



HUGHES INFORMATION TECHNOLOGY SYSTEMS

## ERRATA NOTICE

### EOS Core System (ECS) Project Contract No. NAS5-60000

**January 5, 1998**

**Document No.:** 311-CD-102-001

**Title:** Data Management Database Design and Schema Specifications for the ECS Project

Enclosed please find change pages for the subject document. Please replace the pages as follows:

Remove

4-3 and 4-4

Insert

4-3 and 4-4

If you have any questions, please contact our Data Management Office at (301) 925-0509.

311-CD-102-001

## **EOSDIS Core System Project**

# **Data Management Database Design and Schema Specifications for the ECS Project**

Draft

January 1998

**This document has not yet been approved by the  
Government for general use or distribution.**

Raytheon Systems Company  
Upper Marlboro, Maryland

# **Data Management Database Design and Schema Specifications for the ECS Project**

**Draft**

**January 1998**

Prepared Under Contract NAS5-60000  
CDRL Item #050

## **RESPONSIBLE ENGINEER**

<u>Mary S. Armstrong /s/ for</u>	12/31/97
Maureen Muganda	Date
EOSDIS Core System Project	

## **SUBMITTED BY**

<u>T. W. Fisher /s/</u>	12/31/97
Terry Fisher, ECS CCB Chairman	Date
EOSDIS Core System Project	

**Raytheon Systems Company**  
Upper Marlboro, Maryland

This page intentionally left blank.

# Preface

---

This document describes the data design and database specification for the Data Management subsystem. It is one of ten documents comprising the detailed database design specifications for each of the ECS subsystems.

The subsystem database design specifications for the as delivered system include:

311-CD-100 Access Control (ACL) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-101 Data Distribution (DDIST) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-102 Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-103 Ingest Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-104 Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

311-CD-105 Management Support Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project

311-CD-106 Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project

311-CD-107 Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-108 Storage Management (STMGMT) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-109 Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 1. It requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once approved, contractor approved changes will be handled in accordance with Class I and class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by Document Change Notice (DCN) or by complete revision.

Entity Relationship Diagrams (ERDs) presented in this document have been exported directly from tools and some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on-line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (ECS) on the world-wide web at <http://edhs1.gsfc.nasa.gov>.

Any questions should be addressed to:

Data Management Office  
The ECS Project Office  
Raytheon Systems Company  
1616 McCormick Drive  
Upper Marlboro, MD 20774-5372

## **Abstract**

---

This document outlines “as-built” database design and database schema of the DM database including the physical layout of the database and initial installation parameters.

**Keywords:** data, database, design, configuration, database installation, scripts, security, data model, data dictionary, replication, performance tuning, SQL server, database security, replication, database scripts

This page intentionally left blank.

# Change Information Page

---

List of Effective Pages	
Page Number	Issue
Title	Draft
iii through xii	Draft
1-1 through 1-2	Draft
2-1 through 2-2	Draft
3-1 through 3-2	Draft
4-1 through 4-160	Draft
5-1 through 5-2	Draft
6-1 through 6-2	Draft
7-1 through 7-2	Draft
8-1 through 8-2	Draft
AB-1 through AB-8	

Document History			
Document Number	Status/Issue	Publication Date	CCR Number
311-CD-102-001	Draft	January 1998	97-1755

This page intentionally left blank.

# **Contents**

---

## **Preface**

## **Abstract**

### **1. Introduction**

1.1 Identification .....	1-1
1.2 Scope .....	1-1
1.3 Purpose.....	1-1
1.4 Audience .....	1-1

### **2. Related Documents**

2.1 Applicable Documents.....	2-1
2.2 Information Documents .....	2-1

### **3. Database Configurations**

3.1 Server Configurations .....	3-1
3.2 Storage Device Layouts .....	3-1

### **4. Data Design**

4.1 Database Overview .....	4-1
4.1.1 Physical Data Model Entity Relationship Diagram .....	4-1
4.1.2 Tables .....	4-5
4.1.3 Columns .....	4-32
4.1.4 Column Domains .....	4-53
4.1.5 Rules.....	4-54
4.1.6 Defaults .....	4-54

4.1.7 Views .....	4-54
4.1.8 Integrity Constraints.....	4-56
4.1.9 Triggers .....	4-63
4.1.10 Stored Procedures .....	4-130
4.2 File Usage .....	4-160
4.2.1 Files Definitions.....	4-160
4.2.2 Attributes.....	4-160
4.2.3 Attribute Domains.....	4-160

## **5. Performance and Tuning Factors**

5.1 Indexes .....	5-1
5.2 Segments .....	5-1
5.3 Named Caches.....	5-1

## **6. Database Security**

6.1 Initial Users.....	6-1
6.2 Groups.....	6-1
6.3 Roles .....	6-1
6.4 Object Permissions.....	6-2

## **7. Replication**

7.1 Replication Overview .....	7-1
7.2 Replication Definitions .....	7-1
7.3 Replication Subscriptions .....	7-1
7.4 Replication Database Configuration .....	7-1
7.5 Replication Server Configuration .....	7-1

## **8. Scripts**

8.1 Installation Scripts.....	8-1
8.2 De-Installation Scripts.....	8-1
8.3 Backup/Recovery Scripts.....	8-1
8.4 Miscellaneous Scripts .....	8-1

## **List of Figures**

4-1. ERD Key.....	4-1
4-2. Data Management ERD.....	4-3

## **Abbreviations and Acronyms**

This page is intentionally left blank.

# **1. Introduction**

---

## **1.1 Identification**

This Data Management (DM) Database Design and Database Schema Specification document, Contract Data Requirement List (CDRL) Item Number 050, whose requirements are specified in Data Item description (DID) 311/DV1, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

## **1.2 Scope**

The DM Database Design and Database Schema Specification document describes the data design and database specifications to support the data requirements of Release 2 Drop 3 DM software.

## **1.3 Purpose**

The purpose of the DM Database Design and Database Schema Specification document is to support the maintenance of DM data and databases throughout the life cycle of ECS. This document communicates the database implementation in sufficient detail to support ongoing configuration management.

## **1.4 Audience**

This document is intended to be used by ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope and audience of this document.

Section 2 provides a listing of the related documents, which were used as a source of information for this document.

Section 3 provides a mapping data bases to hardware components.

Section 4 contains the DM physical data model which is the database tables, triggers, stored procedures, and flat files.

Section 5. provides a description of database performance and tuning features such as indexes, caches, and data segments.

Section 6 provides a description of the security infrastructure used and list of the users, groups, and permissions available upon initial installation.

Section 7 contains replication design and implementation details.

Section 8 provides a description of database and database related scripts used for installation, de-installation, backup/recovery, and other miscellaneous functions.

## **2. Related Documents**

---

### **2.1 Applicable Documents**

The following documents, including Internet links, are referenced in the DM Database Design and Database Schema Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume. Internet links cannot be guaranteed for accuracy or currency.

305-CD-100-001	Version 2.0 Segment/Design Specification for the ECS Project
920-TDG-009	Release B0 GSFC DAAC Database Information
920-TDN- 009	Release B0 NSIDC DAAC Database Information
920-TDE-009	Release B0 EDC DAAC Database Information
920-TDL-009	Release B0 LARC DAAC Database Information
920-TDS-009	Release B0 SMC DAAC Database Information
920-TDM-009	Release B0 Mini-DAAC Database Information
922-TDG-XXX	Release B0 DM Disk Partitions
922-TDN-XXX	Release B0 DM Disk Partitions
922-TDE-XXX	Release B0 DM Disk Partitions
922-TDL-XXX	Release B0 DM Disk Partitions
922-TDS-XXX	Release B0 DM Disk Partitions
922-TDM-XXX	Release B0 DM Disk Partitions

### **2.2 Information Documents**

The following documents, although not directly applicable, amplify or clarify the information presented in this document. These documents are not binding on this document.

TBS

This page intentionally left blank.

### **3. Database Configurations**

---

#### **3.1 Server Configurations**

The database configuration of the DM server varies from DAAC to DAAC based on individualized DAAC requirements and hardware availability. These DAAC-specific database configurations are detailed on the following documents:

- 920-TDG-009 Release B0 GSFC DAAC Database Information
- 920-TDN- 009 Release B0 NSIDC DAAC Database Information
- 920-TDE-009 Release B0 EDC DAAC Database Information
- 920-TDL-009 Release B0 LARC DAAC Database Information
- 920-TDS-009 Release B0 SMC DAAC Database Information
- 920-TDM-009 Release B0 Mini-DAAC Database Information

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL <http://pete.hitc.com/baseline/>.

#### **3.2 Storage Device Layouts**

Storage Device layouts, disk partitions, vary from DAAC to DAAC based on the amount of data storage expected to be needed to accommodate a particular DAAC's storage requirements. Disk partitions for the DM server at each DAAC is detailed in the following documents:

- 922-TDG-XXX Release B0 DM Disk Partitions
- 922-TDN-XXX Release B0 DM Disk Partitions
- 922-TDE-XXX Release B0 DM Disk Partitions
- 922-TDL-XXX Release B0 DM Disk Partitions
- 922-TDS-XXX Release B0 DM Disk Partitions
- 922-TDM-XXX Release B0 DM Disk Partitions

These documents are maintained as part of the ECS baseline and available on the world-wide web at the URL <http://pete.hitc.com/baseline/>.

This page intentionally left blank.

## 4. Data Design

---

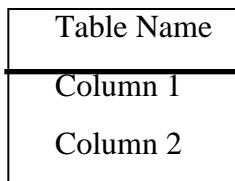
### 4.1 Database Overview

The DM database implements the large majority of the persistent data requirements for the DM subsystem. The database is designed in such a manner as to satisfy business policy while maintaining data integrity and consistency. Database tables are implemented using the Sybase Relational Database Management system (DBMS) version 11.0.1. All components of the DM database are described in the section which follow in sufficient detail to support maintenance needs.

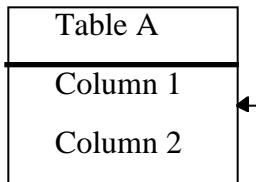
#### 4.1.1 Physical Data Model Entity Relationship Diagram

The Entity Relationship Diagram(ERD) presents a schematic depiction of the DM physical data model. The ERDs presented here for the DM database were produced using the S-Designor Data Architect Computer Aided Software Engineering (CASE) tool. ERDs represent the relationship between entities or database tables. On ERDs, tables are represented by rectangles and relationships are represented as arrow (see Figure 4-1).

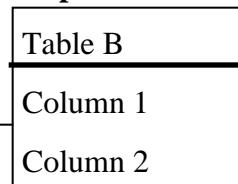
##### Sample Table



##### Independent Table



##### Dependent Table

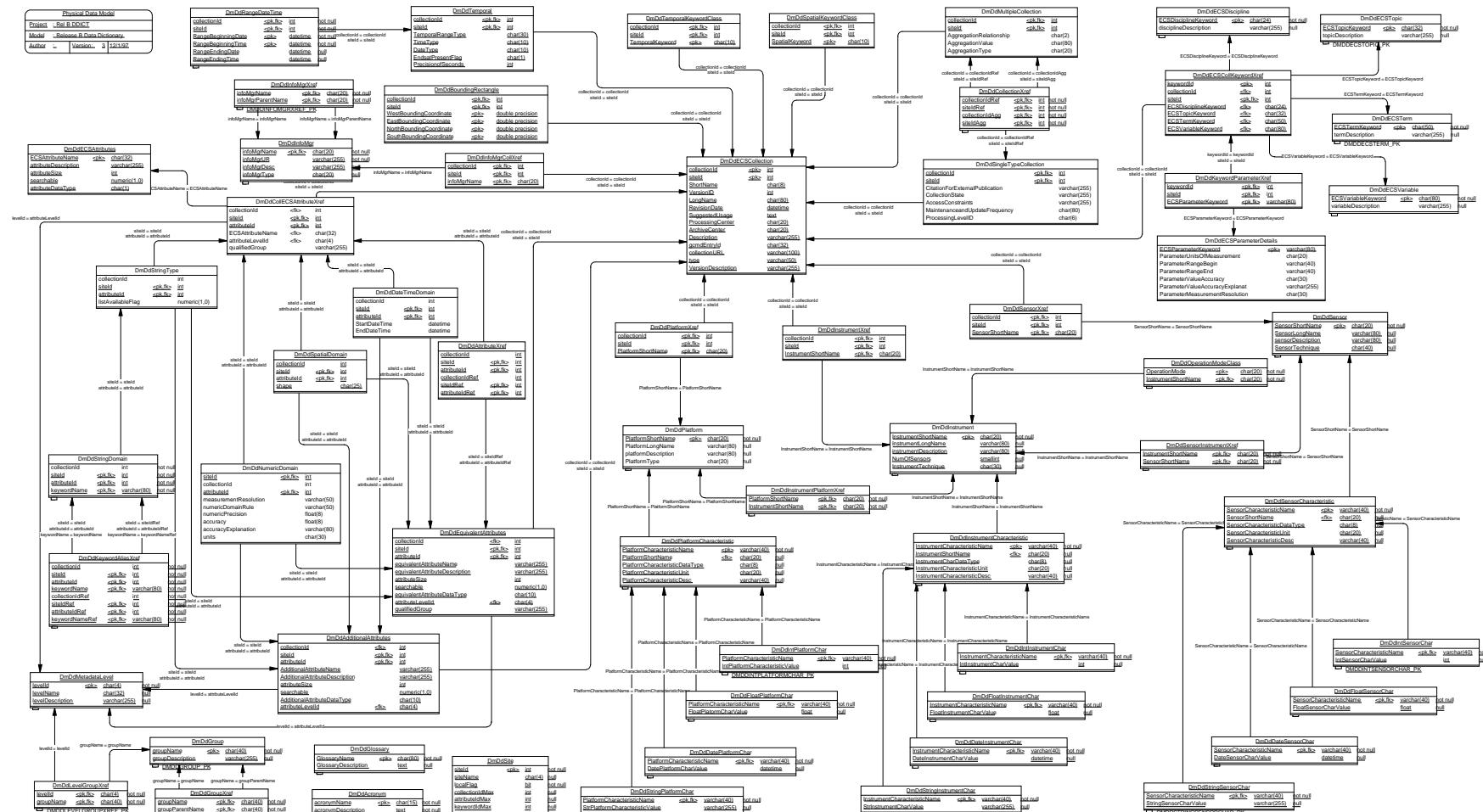


*Table A has zero-to-many relationship with Table B*

**Figure 4-1. ERD Key**

The ERDs for the DM database are shown in Figures 4-1 and 4-2.

This page intentionally left blank.



**Figure 4-2. Data Management ERD**

#### **4.1.2 Tables**

A listing of each the tables in the DM database is given here. A brief definition of each of these tables follows including a listing of the columns comprising the table. The Column List indicates, the column name, the code for the column used in the database, if the column is part of the primary key for the table. That is if the columns can be used alone or in combination with other primary key columns to uniquely identify a single row in the table. The column list also indicates whether the column is a mandatory attribute that must be included in every row.

**Table: DMDDACRONYM****Description**

This table lists all the acronyms that exist.

**Column List**

Name	Code	Type	P	M
acronymDescription	ACRONYMDDESCRIPTION	text	No	No
acronymName	ACRONYMNAME	char(15)	Yes	Yes

**Table: DmDdAdditionalAttributes****Description**

This table identifies the product specific attributes.

**Column List**

Name	Code	Type	P	M
AdditionalAttributeDataType	ADDITIONALATTRIBUTEDATATYPE	char(10)	No	No
AdditionalAttributeDescription	ADDITIONALATTRIBUTEDESCRIPTION	varchar(255)	No	No
AdditionalAttributeName	ADDITIONALATTRIBUTENAME	varchar(255)	No	Yes
attributeId	ATTRIBUTEID	int	Yes	Yes
attributeLevelId	ATTRIBUTELEVELID	char(4)	No	No
attributeSize	ATTRIBUTESIZE	int	No	No
collectionId	COLLECTIONID	int	No	Yes
searchable	SEARCHABLE	numeric(1,0)	No	No
sitId	SITEID	int	Yes	Yes

**Table: DmDdAttributeXref****Description**

This is the a cross reference table between, it holds the attributes, collections and sites references.

**Column List**

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
attributeldRef	ATTRIBUTEIDREF	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
collectionIdRef	COLLECTIONIDREF	int	No	Yes
siteld	SITEID	int	Yes	Yes
siteldRef	SITEIDREF	int	Yes	Yes

**Table: DmDdBoundingRectangle****Description**

This table contains area coverage for ECS collection or granules. This area coverage is expressed by latitude and longitude values in the order western, eastern, northern, and southern - most. For data sets that include a complete band of latitude around the Earth, the west coord = -180.0 and east= 180.0 Latitude values are -90 to +90.

**Column List**

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
EastBoundingCoordinate	EASTBOUNDINGCOORDINATE	double precision	Yes	Yes
NorthBoundingCoordinate	NORTHBOUNDINGCOORDINATE	double precision	Yes	Yes
siteld	SITEID	int	Yes	Yes
SouthBoundingCoordinate	SOUTHBOUNDINGCOORDINATE	double precision	Yes	Yes
WestBoundingCoordinate	WESTBOUNDINGCOORDINATE	double precision	Yes	Yes

**Table: DmDdCollECSAttributeXref****Description**

This table is a cross reference between the ECS attributes and collections.

**Column List**

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
ECSAttributeName	ECSATTRIBUTENAME	char(32)	No	Yes
siteld	SITEID	int	Yes	Yes

**Table: DmDdCollectionXref****Description**

This is a cross reference table between collection and the sites.

**Column List**

Name	Code	Type	P	M
collectionIdAgg	COLLECTIONIDAGG	int	Yes	Yes
collectionIdRef	COLLECTIONIDREF	int	Yes	Yes
siteldAgg	SITEIDAGG	int	Yes	Yes
siteldRef	SITEIDREF	int	Yes	Yes

**Table: DmDdDateInstrumentChar****Description**

This holds the instrument characteristics by date.

## **Column List**

Name	Code	Type	P	M
DateInstrumentCharValue	DATEINSTRUMENTCHARVALUE	datetime	No	No
InstrumentCharacteristicName	INSTRUMENTCHARACTERISTICNAME	varchar(40)	Yes	Yes

## **Table: DmDdDatePlatformChar**

### **Description**

This holds the platform characteristics by date.

## **Column List**

Name	Code	Type	P	M
DatePlatformCharValue	DATEPLATCHARVALUE	datetime	No	No
PlatformCharacteristicName	PLATFORMCHARACTERISTICNAME	varchar(40)	Yes	Yes

## **Table: DmDdDateSensorChar**

### **Description**

This table holds the sensor characteristic by date.

## **Column List**

Name	Code	Type	P	M
DateSensorCharValue	DATESENSORCHARVALUE	datetime	No	No
SensorCharacteristicName	SENSORCHARACTERISTICNAME	varchar(40)	Yes	Yes

## **Table: DmDdDateTimeDomain**

### **Description**

This table holds the date and time domain that a collection can have.

## Column List

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
EndDateTime	ENDDATETIME	datetime	No	No
sitelid	SITEID	int	Yes	Yes
StartTime	STARTDATETIME	datetime	No	Yes

**Table: DmDdECSAttributes**

### Description

This table holds the ECS attributes.

## Column List

Name	Code	Type	P	M
attributeDataType	ATTRIBUTEDATATYPE	char(1)	No	No
attributeDescription	ATTRIBUTEDESCRIPTION	varchar(255)	No	No
attributeLevelId	ATTRIBUTELEVELID	char(4)	No	No
attributeSize	ATTRIBUTESIZE	int	No	No
ECSAttributeName	ECSATTRIBUTENAME	char(32)	Yes	Yes
searchable	SEARCHABLE	numeric(1,0)	No	No

**Table: DmDdECSCollection**

### Description

This table provides further description of the collection to include media, revision date, usage, and processing and archive centers. It is associated with many other collection level descriptive tables.

## Column List

Name	Code	Type	P	M
ArchiveCenter	ARCHIVECENTER	char(20)	No	Yes
collectionId	COLLECTIONID	int	Yes	Yes
collectionURL	COLLECTIONURL	varchar(100)	No	No
Description	DESCRIPTION	varchar(255)	No	No
gcmdEntryId	GCMDENTRYID	char(32)	No	No
LongName	LONGNAME	char(80)	No	No
ProcessingCenter	PROCESSINGCENTER	char(20)	No	No
RevisionDate	REVISIONDATE	datetime	No	No
ShortName	SHORTNAME	char(8)	No	Yes
sitId	SITEID	int	Yes	Yes
SuggestedUsage	SUGGESTEDUSAGE	text	No	No
VersionID	VERSIONID	int	No	Yes
VersionDescription	VERSIONDESCRIPTION	varchar(255)	No	No

**Table: DmDdECSCollKeywordXref**

## Description

This is a cross reference table between Collections and Keywords.

## Column List

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	No	Yes
ECSDisciplineKeyword	ECSDISCIPLINEKEYWORD	char(24)	No	No
ECSTermKeyword	ECSTERMKEYWORD	char(50)	No	No
ECSTopicKeyword	ECSTOPICKEYWORD	char(32)	No	No
ECSVariableKeyword	ECSVARIABLEKEYWORD	char(80)	No	No
keywordId	KEYWORDID	int	Yes	Yes
sitId	SITEID	int	Yes	Yes

**Table: DmDdECSDiscipline****Description**

This table provides the discipline keyword(s) associated with a collection.

**Column List**

Name	Code	Type	P	M
disciplineDescription	DISCIPLINEDESCRIPTION	varchar(255)	No	No
ECSDisciplineKeyword	ECSDISCIPLINEKEYWORD	char(24)	Yes	Yes

**Table: DmDdECSParameterDetails****Description**

This table is used to provide further information about the physical or geophysical parameters specified in the DmDdAdditionalAttributes. It contains the units of measurement, range, accuracy, explanation and resolution.

**Column List**

Name	Code	Type	P	M
ECSParameterKeyword	PARAMETERKEYWORD	varchar(80)	Yes	Yes
ParameterMeasurementResolution	PARAMETERMEASUREMENTRESOLUTION	char(30)	No	No
ParameterRange	PARAMETERRANGE	char(10)	No	No
ParameterUnitsOfMeasurement	PARAMETERUNITSOFMEASUREMENT	char(20)	No	No
ParameterValueAccuracy	PARAMETERVALUEACCURACY	char(30)	No	No
ParameterValueAccuracyExplanat	PARAMETERVALUEACCURACYEXPLANAT	varchar(255)	No	No

**Table: DmDdECSTerm****Description**

This table contains the term keyword(s) associated with the collection. (i.e., atmospheric temperature, precipitation, soils, sea ice).

### **Column List**

Name	Code	Type	P	M
ECSTermKeyword	ECSTERMKEYWORD	char(50)	Yes	Yes
termDescription	TERMDESCRIPTION	varchar(255)	No	No

**Table: DmDdECSTopic**

### **Description**

This table contains the topic keyword(s) associated with the collection.(i.e., atmospheric science, hydrosphere, land surface, ocean science).

### **Column List**

Name	Code	Type	P	M
ECSTopicKeyword	ECSTOPICKEYWORD	char(32)	Yes	Yes
topicDescription	TOPICDESCRIPTION	varchar(255)	No	No

**Table: DmDdECSVariable**

### **Description**

This table contains the variable keyword(s) associated with the collection. (i.e., upper troposphere temperature, precipitable water, soil depth, albedo).

### **Column List**

Name	Code	Type	P	M
ECSVariableKeyword	ECSVARIABLEKEYWORD	char(80)	Yes	Yes
variableDescription	VARIABLEDESCRIPTION	varchar(255)	No	No

**Table: DmDdEquivalentAttributes****Description**

This table holds extra information for equivalent attributes found in ECS.

**Column List**

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
attributeSize	ATTRIBUTESIZE	int	No	No
collectionId	COLLECTIONID	int	No	Yes
equivalentAttributeDataType	ADDITIONALATTRIBUTEDATATYPE	char(10)	No	No
equivalentAttributeDescription	ADDITIONALATTRIBUTEDESCRIPTION	varchar(255)	No	No
equivalentAttributeName	EQUIVALENTATTRIBUTENAME	varchar(255)	No	Yes
searchable	SEARCHABLE	numeric(1,0)	No	No
sited	SITEID	int	Yes	Yes

**Table: DmDdFloatInstrumentChar****Description**

This table holds information for the instrument characteristics that have float as a data type.

**Column List**

Name	Code	Type	P	M
FloatInstrumentCharValue	FLOATINSTRUMENTCHARVALUE	float	No	No
InstrumentCharacteristicName	INSTRUMENTCHARACTERISTICNAME	varchar(40)	Yes	Yes

**Table: DmDdFloatPlatformChar****Description**

This table holds information for the platform characteristics that have float as a data type.

## Column List

Name	Code	Type	P	M
FloatPlatormCharValue	FLOATPLATORMCHARVALUE	float	No	No
PlatformCharacteristicName	PLATFORMCHARACTERISTICNAME	varchar(40)	Yes	Yes

**Table: DmDdFloatSensorChar**

## Description

This table holds information for the sensor characteristics that have float as a data type.

## Column List

Name	Code	Type	P	M
FloatSensorCharValue	FLOATSENSORCHARVALUE	float	No	No
SensorCharacteristicName	SENSORCHARACTERISTICNAME	varchar(40)	Yes	Yes

**Table: DmDdGlossary**

## Description

This holds the glossary information of the ECS.

## Column List

Name	Code	Type	P	M
GlossaryDescription	GLOSSARYDESCRIPTION	text	No	No
GlossaryName	GLOSSARYNAME	char(80)	Yes	Yes

**Table: DmDdGroup**

## Description

This table holds the group within the ESDT.

## Column List

Name	Code	Type	P	M
groupDescription	GROUPDESCRIPTION	varchar(255)	No	No
groupName	GROUPNAME	char(40)	Yes	Yes

**Table: DmDdGroupXref**

## Description

This table shows the relationship between the groups.

## Column List

Name	Code	Type	P	M
groupName	GROUPNAME	char(40)	Yes	Yes
groupParentName	GROUPPARENTNAME	char(40)	Yes	Yes

**Table: DmDdInfoMgr**

## Column List

Name	Code	Type	P	M
infoMgrDesc	INFOMGR	varchar(255)	No	No
infoMgrName	INFOMGRNAME	char(20)	Yes	Yes
infoMgrType	INFOMGRTYPE	char(20)	No	No
infoMgrUR	INFOMGRUR	varchar(255)	No	Yes

**Table: DmDdInfoMgrCollXref**

## Description

This table is a cross reference between InfoMgr and Collection.

## Column List

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
infoMgrName	INFOMGRNAME	char(20)	Yes	Yes
sited	SITEID	int	Yes	Yes

**Table: DmDdInfoMgrXref**

## Description

This table shows the relationship between the InfoMgr.

## Column List

Name	Code	Type	P	M
infoMgrName	INFOMGRNAME	char(20)	Yes	Yes
infoMgrParentName	INFOMGRPARENTNAME	char(20)	Yes	Yes

**Table: DmDdInstrument**

## Description

This class defines the device used to measure or record data, including direct human observation. Included in this class are defined EOS Instruments. In cases where instruments have a single sensor or the instrument and sensor are used synonomously (e.g. AVHRR) the both Instrument and sensor should be recorded.

## Column List

Name	Code	Type	P	M
instrumentDescription	INSTRUMENTDESCRIPTION	varchar(80)	No	No
InstrumentLongName	INSTRUMENTLONGNAME	varchar(80)	No	No
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	Yes	Yes
InstrumentTechnique	INSTRUMENTTECHNIQUE	char(30)	No	No
NumOfSensors	NUMOFSENSORS	smallint	No	No

**Table: DmDdInstrumentCharacteristic****Description**

This class is used to define the characteristics of instrument specific attributes. It should not be used to define attributes of new objects.

**Column List**

Name	Code	Type	P	M
InstrumentCharacteristicDesc	INSTRUMENTCHARACTERISTICDESC	varchar(40)	No	No
InstrumentCharacteristicName	INSTRUMENTCHARACTERISTICNAME	varchar(40)	Yes	Yes
InstrumentCharacteristicUnit	INSTRUMENTCHARACTERISTICUNIT	char(20)	No	No
InstrumentCharDataType	INSTRUMENTCHARDATATYPE	char(6)	No	No
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	No	No

**Table: DmDdInstrumentPlatformXref****Description**

This is a cross reference table between instrument and platform.

**Column List**

Name	Code	Type	P	M
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	Yes	Yes
PlatformShortName	PLATFORMSHORTNAME	char(20)	Yes	Yes

**Table: DmDdInstrumentXref****Description**

This is a cross reference table between instrument and collection.

### **Column List**

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	Yes	Yes
sited	SITEID	int	Yes	Yes

**Table: DmDdIntInstrumentChar**

### **Description**

This table holds information for the instrument characteristics that have integer as a data type.

### **Column List**

Name	Code	Type	P	M
InstrumentCharacteristicName	INSTRUMENTCHARACTERISTICNAME	varchar(40)	Yes	Yes
IntInstrumentCharValue	INTINSTRUMENTCHARVALUE	int	No	No

**Table: DmDdIntPlatformChar**

### **Description**

This table holds information for the platform characteristics that have integer as a data type.

### **Column List**

Name	Code	Type	P	M
IntPlatformCharacteristicValue	INTPLATFORMCHARACTERISTICVALUE	int	No	No
PlatformCharacteristicName	PLATFORMCHARACTERISTICNAME	varchar(40)	Yes	Yes

**Table: DmDdIntSensorChar**

### **Description**

This table holds information for the sensor characteristics that have integer as a data type.

### **Column List**

Name	Code	Type	P	M
IntSensorCharValue	INTSENSORCHARVALUE	int	No	No
SensorCharacteristicName	SENSORCHARACTERISTICNAME	varchar(40)	Yes	Yes

**Table: DmDdKeywordAliasXref**

### **Description**

This table holds the aliases for a keyword.

### **Column List**

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
attributeldRef	ATTRIBUTEIDREF	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
collectionIdRef	COLLECTIONIDREF	int	No	Yes
keywordName	KEYWORDNAME	varchar(80)	Yes	Yes
keywordNameRef	KEYWORDNAMEREF	varchar(80)	Yes	Yes
siteld	SITEID	int	Yes	Yes
siteldRef	SITEIDREF	int	Yes	Yes

**Table: DmDdKeywordParameterXref**

### **Description**

This is a cross reference table between ECS keywords parameters and the DDICT parameter keywords.

### **Column List**

Name	Code	Type	P	M
ECSParameterKeyword	ECSPARAMETERKEYWORD	varchar(80)	Yes	Yes
keywordId	KEYWORDID	int	Yes	Yes
sited	SITEID	int	Yes	Yes

**Table: DmDdLevelGroupXref**

### **Description**

This table is the cross reference that shows the relationship to which groups can belong to which levels.

### **Column List**

Name	Code	Type	P	M
groupName	GROUPNAME	char(40)	Yes	Yes
levelId	LEVELID	char(4)	Yes	Yes

**Table: DmDdMetadataLevel**

### **Description**

This table defines the level that attribute appears in collection or granule.

### **Column List**

Name	Code	Type	P	M
levelDescription	LEVELDESCRIPTION	varchar(255)	No	No
levelId	LEVELID	char(4)	Yes	Yes
levelName	LEVELNAME	char(32)	No	No

**Table: DmDdMultipleCollection****Description**

This table holds the multiple collection information.

**Column List**

Name	Code	Type	P	M
AggregationRelationship	AGGREGATIONRELATIONSHIP	char(2)	No	No
AggregationType	AGGREGATIONTYPE	char(20)	No	No
AggregationValue	AGGREGATIONVALUE	char(80)	No	No
collectionId	COLLECTIONID	int	Yes	Yes
sitedId	SITEID	int	Yes	Yes

**Table: DmDdNumericDomain****Description**

This table holds the numeric domain for a collection.

**Column List**

Name	Code	Type	P	M
accuracy	ACCURACY	float(8)	No	No
accuracyExplanation	ACCURACYEXPLANATION	varchar(80)	No	No
attributefId	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
measurementResolution	MEASUREMENTRESOLUTION	varchar(50)	No	Yes
numericDomainRule	NUMERICDOMAINRULE	varchar(50)	No	No
numericPrecision	NUMERICPRECISION	float(8)	No	No
sitedId	SITEID	int	Yes	Yes
units	UNITS	char(30)	No	No

**Table: DmDdOperationModeClass****Column List**

Name	Code	Type	P	M
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	No	No
OperationMode	OPERATIONMODE	char(20)	Yes	Yes

**Table: DmDdPlatform****Description**

This class describes the relevant platforms associated with the acquisition of the collection or granule. Platform types include Spacecraft, Aircraft, Vessel, Buoy, Platform, Station, Network or Human. In cases where Human is the platform type it should be of scientific relevancy to the collection. If an instrument is hand held and that is relevant to the collection of the data then PlatformType=Human. In cases where an instrument is hand-held but the human is associated with another platform then all relevant platforms should be associated with the collection. Humans can be both Platforms and Instruments (e.g. if a human is standing on the ground and makes a visual observation then: PlatformType=Human, Instrument=HumanObservation, SensorShortName=HumanVisual).

**Column List**

Name	Code	Type	P	M
platformDescription	PLATFORMDESCRIPTION	varchar(80)	No	No
PlatformLongName	PLATFORMLONGNAME	varchar(80)	No	No
PlatformShortName	PLATFORMSHORTNAME	char(20)	Yes	Yes
PlatformType	PLATFORMTYPE	char(20)	No	No

**Table: DmDdPlatformCharacteristic****Description**

This class is used to define the characteristics of platform specific attributes. It should not be used to define attributes of new objects.

## Column List

Name	Code	Type	P	M
PlatformCharacteristicDataType	PLATFORMCHARACTERISTICDATATYPE	char(6)	No	No
PlatformCharacteristicDesc	PLATFORMCHARACTERISTICDESC	varchar(40)	No	No
PlatformCharacteristicName	PLATFORMCHARACTERISTICNAME	varchar(40)	Yes	Yes
PlatformCharacteristicUnit	PLATFORMCHARACTERISTICUNIT	char(20)	No	No
PlatformShortName	PLATFORMSHORTNAME	char(20)	No	No

**Table: DmDdPlatformXref**

## Description

This is cross reference table between platform and collection.

## Column List

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
PlatformShortName	PLATFORMSHORTNAME	char(20)	Yes	Yes
sitedId	SITEID	int	Yes	Yes

**Table: DmDdRangeDateTime**

## Description

This class specifies the start and end date/time of a granule or collection.

## Column List

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
RangeBeginningDate	RANGEBEGINNINGDATE	datetime	Yes	Yes
RangeBeginningTime	RANGEBEGINNINGTIME	datetime	Yes	Yes
RangeEndingDate	RANGEENDINGDATE	datetime	No	No
RangeEndingTime	RANGEENDINGTIME	datetime	No	No
sitelid	SITEID	int	Yes	Yes

**Table: DmDdSensor**

## Description

This class is used to describe sensory subcomponents of an instrument. In cases where instruments have a single sensor or the Instrument and Sensor are used synonomously (e.g. AVHRR) both the Instrument and Sensor should be recorded.

## Column List

Name	Code	Type	P	M
sensorDescription	SENSORDESCRIPTION	varchar(80)	No	No
SensorLongName	SENSORLONGNAME	varchar(80)	No	No
SensorShortName	SENSORSHORTNAME	char(20)	Yes	Yes
SensorTechnique	SENSORTECHNIQUE	char(40)	No	No

**Table: DmDdSensorCharacteristic**

## Description

This class is used to define the characteristics of sensor specific attributes. It should not be used to define attributes of new objects.

## Column List

Name	Code	Type	P	M
SensorCharacteristicDataType	SENSORCHARACTERISTICDATATYPE	char(6)	No	No
SensorCharacteristicDesc	SENSORCHARACTERISTICDESC	varchar(40)	No	No
SensorCharacteristicName	SENSORCHARACTERISTICNAME	varchar(40)	Yes	Yes
SensorCharacteristicUnit	SENSORCHARACTERISTICUNIT	char(20)	No	No
SensorShortName	SENSORSHORTNAME	char(20)	No	No

**Table: DmDdSensorInstrumentXref**

## Description

This is a cross reference table between sensors and instruments.

## Column List

Name	Code	Type	P	M
InstrumentShortName	INSTRUMENTSHORTNAME	char(20)	Yes	Yes
SensorShortName	SENSORSHORTNAME	char(20)	Yes	Yes

**Table: DmDdSensorXref**

## Description

This is a cross reference between sensor and collection.

## Column List

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
SensorShortName	SENSORSHORTNAME	char(20)	Yes	Yes
sitelId	SITEID	int	Yes	Yes

**Table: DmDdSingleTypeCollection****Description**

This class provides a description specific to a single, as opposed to a multitype collection, to include citation of external publication, collection state, maintenance and update frequency, and access constraints. The definition of a singletype collection is stated below. The management and development of singletype collections is the subject of other documentation.

A single type collection contains a set of granules for which the dominant variation in the value of metadata attributes is in the space and time attributes.

**Column List**

Name	Code	Type	P	M
AccessConstraints	ACCESSCONSTRAINTS	varchar(255)	No	No
CitationForExternalPublication	CITATIONFOREXTERNALPUBLICATION	varchar(255)	No	No
collectionId	COLLECTIONID	int	Yes	Yes
CollectionState	COLLECTIONSTATE	varchar(255)	No	No
MaintenanceandUpdateFrequency	MAINTENANCEANDUPDATEFREQUENCY	char(80)	No	No
ProcessingLevelID	PROCESSINGLEVELID	char(6)	No	Yes
siteld	SITEID	int	Yes	Yes

**Table: DmDdSite****Description**

This table holds information about the site. i.e., site name, site id, etc.

### **Column List**

Name	Code	Type	P	M
attributeIdMax	ATTRIBUTEID	int	No	No
collectionIdMax	COLLECTIONID	int	No	No
keywordIdMax	KEYWORDIDMAX	int	No	No
localFlag	LOCALFLAG	bit	No	Yes
siteld	SITEID	int	Yes	Yes
siteName	SITENAME	char(4)	No	No

**Table: DmDdSpatialDomain**

### **Description**

This table holds spatial information about a collection.

### **Column List**

Name	Code	Type	P	M
attributeId	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
shape	SHAPE	char(25)	No	No
siteld	SITEID	int	Yes	Yes

**Table: DmDdSpatialKeywordClass**

### **Description**

This class contains the spatial keywords associated with the ECS collection.

### **Column List**

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
sited	SITEID	int	Yes	Yes
SpatialKeyword	SPATIALKEYWORD	char(10)	Yes	Yes

**Table: DmDdStringDomain**

### **Description**

TBS

### **Column List**

Name	Code	Type	P	M
attributeld	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
keywordName	STRINGKEYWORDNAME	varchar(80)	Yes	Yes
sited	SITEID	int	Yes	Yes

**Table: DmDdStringInstrumentChar**

### **Description**

This table holds information for the instrument characteristics that have string as a data type.

### **Column List**

Name	Code	Type	P	M
InstrumentCharacteristicName	INSTRUMENTCHARACTERISTICNAME	varchar(40)	Yes	Yes
StrInstrumentCharValue	STRINSTRUMENTCHARVALUE	varchar(255)	No	No

**Table: DmDdStringPlatformChar****Description**

This table holds information for the platform characteristics that have string as a data type.

**Column List**

Name	Code	Type	P	M
PlatformCharacteristicName	PLATFORMCHARACTERISTICNAME	varchar(40)	Yes	Yes
StrPlatformCharacteristicValue	STRPLATFORMCHARACTERISTICVALUE	varchar(255)	No	No

**Table: DmDdStringSensorChar****Description**

This table holds information for the sensor characteristics that have string as a data type.

**Column List**

Name	Code	Type	P	M
SensorCharacteristicName	SENSORCHARACTERISTICNAME	varchar(40)	Yes	Yes
StringSensorCharValue	STRINGSENSORCHARVALUE	varchar(255)	No	No

**Error! Bookmark not defined.Table: DmDdStringType****Column List**

Name	Code	Type	P	M
attributelid	ATTRIBUTEID	int	Yes	Yes
collectionId	COLLECTIONID	int	No	Yes
listAvailableFlag	LISTAVALABLEFLAG	numeric(1,0)	No	No
sitelid	SITEID	int	Yes	Yes

**Table: DmDdTemporal****Description**

This class contains attributes which describe the basis of the time system used in other classes

**Column List**

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
DateType	DATETYPE	char(10)	No	No
EndsatPresentFlag	ENDSATPRESENTFLAG	char(1)	No	No
PrecisionofSeconds	PRECISIONOFSECONDS	int	No	No
sitelid	SITEID	int	Yes	Yes
TemporalRangeType	TEMPORALRANGETYPE	char(30)	No	No
TimeType	TIMETYPE	char(10)	No	No

**Table: DmDdTemporalKeywordClass****Description**

This class identifies the type of temporal characterization for a granule or collection.

**Column List**

Name	Code	Type	P	M
collectionId	COLLECTIONID	int	Yes	Yes
sitelid	SITEID	int	Yes	Yes
TemporalKeyword	TEMPORALKEYWORD	char(10)	Yes	Yes

### **4.1.3 Columns**

Brief definitions of each of the columns present in the database tables defined above are contained herein.

#### **Column: ACCESSCONSTRAINTS**

##### **Description**

Restrictions and legal prerequisites for accessing the collection. These include any access constraints applied to assure the protection of privacy or intellectual property, and any special restrictions or limitations on obtaining the collection.

These restrictions differ from Use Restrictions in that they only apply to access.

Valid Values : See 311-CD-107-001

#### **Column: ACCURACY**

##### **Description**

Describes the accuracy of a location as a numerical value. See also ACCURACYEXPLANATION.

Valid Values : See 311-CD-107-001

#### **Column: ACCURACYEXPLANATION**

##### **Description**

Describes what the numeric value of ACCURACY represents.

Valid Values : See 311-CD-107-001

#### **Column: ACRONYMDESCRIPTION**

##### **Description**

This is the acronym description.

#### **Column: ACRONYMNAME**

##### **Description**

This is the acronym name.

#### **Column: ADDITIONALATTRIBUTEDATATYPE**

##### **Description**

This is the Data Type of the ParameterValue.

Valid Values : See 311-CD-107-001

#### **Column: ADDITIONALATTRIBUTEDESCRIPTION**

##### **Description**

This attribute provides a description for the AddtionalAttributeName.

Valid Values : See 311-CD-107-001

#### **Column: ADDITIONALATTRIBUTENAME**

##### **Description**

This is the Data Type of AdditionalAttributeName.

Valid Values : See 311-CD-107-001

#### **Column: AGGREGATIONRELATIONSHIP**

##### **Description**

This attribute identifies the relationship between the aggregation attribute and its corresponding value. This relationship may be expressed as Boolean operations i.e. ‘=, <, >, ne’

Valid Values : See 311-CD-107-001

#### **Column: AGGREGATIONTYPE**

##### **Description**

This attribute will contain the criteria by which multiple type collections have been grouped. It will describe the major categorization which applies to the data therein. Possible collection groupings include: INSTRUMENT, for all collections associated with a given collecting instrument such as CERES—this is a common aggregation criteria for ECS ‘datasets’; PROJECT, for all data associated with a given project that may or may not be related to a single instrument, such as FIRE—this is again a common aggregation criteria for ECS ‘datasets’; PARAMETER, for all granules that reflect measurements of a single specific (or related group of specific) geophysical parameters, such as: CLOUD PROPERTIES—this is often an aggregation criteria for ECS ‘products’; SUPERGRANULE, for collections of granules that the data provider wishes to be orderable as a single related grouping, such as SSM/I TIME SERIES—this is a concept adopted from MSFC use; EVENT for a predetermined/tagged set of granules that have been found to be related to a particular geophysical phenomena or event, such as MIDWEST FLOOD ’93 or OZONE HOLE or MT. PINATUBO—this is a new ECS concept, also suggested by the University of Virginia Atmospheric researchers.

Valid Values : See 311-CD-107-001

**Column: AGGREGATIONVALUE****Description**

This attribute contains the value associated with the aggregation type. An example may be EVENT (aggregation type) = MIDWEST FLOOD '93 (aggregation value). MIDWEST FLOOD '93 would be the value associated with the event or aggregation type.

Valid Values : See 311-CD-107-001

**Column: ARCHIVECENTER****Description**

This is the center where collection is archive.

Valid Values : See 311-CD-107-001

**Column: ATTRIBUTEDATATYPE****Description**

This is the data type of the ECS attribute.

Valid Values : See 311-CD-107-001

**Column: ATTRIBUTEDESCRIPTION****Description**

This holds the ECS attribute description.

Valid Values : See 311-CD-107-001

**Column: ATTRIBUTEID****Description**

This is the attribute identifier.

**Column: ATTRIBUTEIDREF****Description**

This is the attribute identifier reference

**Column: ATTRIBUTELEVELID****Description**

This reflects the classification of the attribute level, which defines in general terms the characteristics of the attribute.

**Column: ATTRIBUTESIZE****Description**

This is the ECS attribute size.

**Column: CITATIONFOREXTERNALPUBLICATION****Description**

The recommended reference to be used when referring to this collection in publications. Its format is free text, but should include: Originator (the name of an organization or individual that developed the data set, where Editor(s)' names are followed by (ed.) and Compiler(s)' names are followed by (comp.)); Publication date (the date of publication or release of the data set); Title (the name by which document can be referenced).

Valid Values : See 311-CD-107-001

**Column: COLLECTIONID****Description**

This is the unique collection identifier.

**Column: COLLECTIONIDREF****Description**

This is the Collection identifier reference.

**Column: COLLECTIONSTATE****Description**

This attribute describes the state of the collection, whether it is planned but not yet existent, partially complete due to continual additions from remotely sensed data/processing/reprocessing, or is considered a complete product/dataset.

Valid Values : See 311-CD-107-001

**Column: COLLECTIONURL****Description**

This is the collection URL.

**Column: DATEINSTRUMENTCHARVALUE****Description**

This holds the date on which the instrument characteristic value was issued.

**Column: DATEPLATCHARVALUE****Description**

This holds the date on which the platform characterisitc value was issued.

**Column: DATESENSORCHARVALUE****Description**

This is the date on which the sensor characteristic value was issued.

**Column: DATETYPE****Description**

This attribute specifies the type of date represented by the value in the date attributes of the temporal subclasses.

Valid Values : See 311-CD-107-001

**Column: DESCRIPTION****Description**

This is the description of a collection.

Valid Values : See 311-CD-107-001

**Column: DISCIPLINEDESCRIPTION****Description**

This is a description for a discipline.

**Column: EASTBOUNDINGCOORDINATE****Description**

Eastern-most the limit of coverage expressed in longitude.

Valid Values : See 311-CD-107-001

**Column: ECSATTRIBUTENAME****Description**

This holds the ECS attribute name.

Valid Values : See 311-CD-107-001

**Column: ECSDISCIPLINEKEYWORD****Description**

This is the keyword used to describe the general discipline area of the collection. A collection can conceivably cover several disciplines.

Valid Values : See 311-CD-107-001

**Column: PARAMETERKEYWORD****Description**

Keyword used to describe specific characteristics of a collection at a higher level of detail than provided by ECSVariableKeyword.

Valid Values : See 311-CD-107-001

**Column: ECSTERMKEYWORD****Description**

Keyword used to describe the science parameter area of the collection. A collection can conceivably cover many such parameters.

Valid Values : See 311-CD-107-001

**Column: ECSTOPICKEYWORD****Description**

Keyword used to describe the general topic area of the collection. A collection can conceivably cover several topics.

Valid Values : See 311-CD-107-001

**Column: ECSVARIABLEKEYWORD****Description**

Keyword used to describe the specific science parameter content of the collection. A collection can conceivably cover many specific parameters. The keyword valids are the lowest level physical parameter terms which are normally searched by a user; i.e. a user enters a keyword which when found may connect with one or more parameters from collections. The keywords are also the lowest level words which describe product content without being the server specific measurement (held in Parameter class). While there is a controlled list of these parameters held by GCMD, additions can be made by an as yet unspecified configuration control process. This is an ECS attribute.

Valid Values : See 311-CD-107-001

**Column: ENDDATETIME****Description**

The ending date of a collection.

Valid Values : See 311-CD-107-001

**Column: ENDSATPRESENTFLAG****Description**

This attribute will denote that a data collection which covers, temporally, a discontinuous range, currently ends at the present date. This way, the granules which comprise the data collection that are continuously being added to inventory need not update the data collection metadata for each one. Note that MODIS granules may be added several thousand times a day, making the update of the data collection metadata impractical.

Valid Values : See 311-CD-107-001

**Column: ADDITIONALATTRIBUTEDATATYPE****Description**

This is the equivalent Data Type of the ParameterValue.

Valid Values : See 311-CD-107-001

**Column: ADDITIONALATTRIBUTEDESCRIPTION****Description**

This equivalent attribute provides a description for the AddtionalAttributeName.

Valid Values : See 311-CD-107-001

**Column: EQUIVALENTATTRIBUTENAME****Description**

TBS

**Column: FLOATINSTRUMENTCHARVALUE****Description**

This holds the float value of the instrument.

**Column: FLOATPLATORMCHARVALUE****Description**

This holds the float value of the platform.

**Column: FLOATSENSORCHARVALUE****Description**

This is the float value of a sensor.

**Column: GCMDENTRYID****Description**

TBS

**Column: GLOSSARYDESCRIPTION****Description**

This is the glossary description.

**Column: GLOSSARYNAME****Description**

This is the glossary name.

**Column: GROUPDESCRIPTION****Description**

This is the group description.

**Column: GROUPNAME****Description**

This is the group name. (i.e., Spatial, SpatialDomainContainer, etc.)

**Column: GROUUPARENTNAME****Description**

The is the name of the group parent. (i.e., for Spatial -- ROOT, for SpatialDomainContainer -- Spatial, etc.)

**Column: INFOMGR****Description**

The description of the information manager.

**Column: INFOMGRNAME****Description**

The name of the information manager.

**Column: INFOMGRPARENTNAME****Description**

Refers to the information manager's parent name.

**Column: INFOMGRTYPE****Description**

States the type of information manager.

**Column: INFOMGRUR****Description**

The information manager's Universal Reference.

**Column: INSTRUMENTCHARACTERISTICDESC****Description**

The description of the instrument attribute.

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTCHARACTERISTICNAME****Description**

The name of the instrument characteristic attribute. Instrument characteristic are instrument-specific attributes.

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTCHARACTERISTICUNIT****Description**

The units of the attribute defined with InstrumentCharacteristic.

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTCHARDATATYPE****Description**

The datatype of the instrument characteristic/attribute defined by InstrumentCharacteristicName.

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTDESCRIPTION****Description**

This is the instrument description.

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTLONGNAME****Description**

The expanded name of the primary sensory instrument. (e.g. Advanced Spaceborne Thermal Emission and Reflective Radiometer, Clouds and the Earth's Radiant Energy System, Human Observation).

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTSHORTNAME****Description**

The unique identifier of an instrument. (e.g. ASTER, AVHRR-3, CERES, Human).

Valid Values : See 311-CD-107-001

**Column: INSTRUMENTTECHNIQUE****Description**

The instrument method or procedure.. (e.g. radiometer, manual enumeration).

Valid Values : See 311-CD-107-001

**Column: INTINSTRUMENTCHARVALUE****Description**

This holds the integer value of the instrument.

**Column: INTPLATFORMCHARACTERISTICVALUE****Description**

This holds the integer value of the platform.

**Column: INTSENSORCHARVALUE****Description**

This holds the integer value of the sensor.

**Column: KEYWORDID****Description**

This is the keyword identifier.

**Column: KEYWORDIDMAX****Description**

TBD

**Column: STRINGKEYWORDNAME****Description**

TBD

**Column: KEYWORDNAME****Description**

This is the keyword name for the DDICT.

**Column: KEYWORDNAMEREF****Description**

This is the keyword name reference.

**Column: LEVELDESCRIPTION****Description**

This is the Level description.

**Column: LEVELID****Description**

This is the level Identifier.

**Column: LEVELNAME****Description**

Valid Values : See 311-CD-107-001

**Column: LISTAVAILABLEFLAG****Description**

TBD

**Column: LOCALFLAG****Description**

TDB

**Column: LONGNAME****Description**

This the collection long name.

Valid Values : See 311-CD-107-001

**Column: MAINTENANCEANDUPDATEFREQUENCY****Description**

The frequency with which changes and additions are made to the collection after the initial dataset begins to be collected/processed.

Valid Values : See 311-CD-107-001

**Column: MEASUREMENTRESOLUTION**

**Description**

Valid Values : See 311-CD-107-001

**Column: NORTHBOUNDINGCOORDINATE**

**Description**

Northern-most coordinate of the limit of coverage expressed in geodetic latitude.

Valid Values : See 311-CD-107-001

**Column: NUMERICDOMAINRULE**

**Description**

TBD

**Column: NUMERICPRECISION**

**Description**

Valid Values : See 311-CD-107-001

**Column: NUMOFSENSORS**

**Description**

The number of discrete (if any) sensors on an instrument.

Valid Values : See 311-CD-107-001

**Column: OPERATIONMODE**

**Description**

valids = launch, survival, initialization, safe, diagnostic, roll, tilt, standby, routine, test, calibration

Valid Values : See 311-CD-107-001

**Column: PARAMETERMEASUREMENTRESOLUTION**

**Description**

This attribute will be used to identify the smallest unit increment to which the parameter value is measured.

Valid Values : See 311-CD-107-001

**Column: PARAMETERRANGE****Description**

This attribute provides maximum and minimum value of parameter over whole collection.

Valid Values : See 311-CD-107-001

**Column: PARAMETERUNITSOFMEASUREMENT****Description**

The standard units of measurement for a non-core attribute. AVHRR: Units of Geophysical Parameter=Units of Geophysical Parameter.

Valid Values : See 311-CD-107-001

**Column: PARAMETERVALUEACCURACY****Description**

An estimate of the accuracy of the assignment of attribute value. i.e. AVHRR: Measurement Error or Precision=Measurement error or precision of a data product parameter. This can be specified in percent or the units with which the parameter is measured.

Valid Values : See 311-CD-107-001

**Column: PARAMETERVALUEACCURACYEXPLANAT****Description**

This defines the method used for determining the Parameter Value Accuracy that is given for this non core attribute.

Valid Values : See 311-CD-107-001

**Column: PLATFORMCHARACTERISTICDATATYPE****Description**

The datatype of the Platform Characteristic/attribute defined by PlatformCharacteristicName.

Valid Values : See 311-CD-107-001

**Column: PLATFORMCHARACTERISTICDESC****Description**

Description of the Platform Characteristic attribute.

Valid Values : See 311-CD-107-001

**Column: PLATFORMCHARACTERISTICNAME****Description**

The name of the Platform Characteristic attribute.

Valid Values : See 311-CD-107-001

**Column: PLATFORMCHARACTERISTICUNIT****Description**

Units associated with the Platform Characteristic attribute value.

Valid Values : See 311-CD-107-001

**Column: PLATFORMDESCRIPTION****Description**

This is the platform description.

Valid Values : See 311-CD-107-001

**Column: PLATFORMLONGNAME****Description**

The expanded or long name of the platform associated with an instrument.

Valid Values : See 311-CD-107-001

**Column: PLATFORMSHORTNAME****Description**

The unique platform name. (e.g. GOES-8).

Valid Values : See 311-CD-107-001

**Column: PLATFROMTYPE****Description**

The most relevant platform type.

Valid Values : See 311-CD-107-001

**Column: PRECISIONOFSECONDS****Description**

The precision (position in number of places to right of decimal point) of seconds used in measurement.

Valid Values : See 311-CD-107-001

**Column: PROCESSINGCENTER****Description**

This is the name of the processing center like one of the DAAC (i.e., GSFC, JPL, etc.) or SCF.

Valid Values : See 311-CD-107-001

**Column: PROCESSINGLEVELID****Description**

This attribute reflects the classification of the science data processing level, which defines in general terms the characteristics of the output of the processing performed.

Valid Values : See 311-CD-107-001

**Column: RANGEBEGINNINGDATE****Description**

The year (and optionally month, or month and day) when the temporal coverage period being described began.

Valid Values : See 311-CD-107-001

**Column: RANGEBEGINNINGTIME****Description**

The first hour (and optionally minute, or minute and second) of the temporal coverage period being described.

Valid Values : See 311-CD-107-001

**Column: RANGEENDINGDATE****Description**

The last year (and optionally month, or month and day) of the temporal coverage period being described.

GSFC AVHRR This date represents the end date of the latest granule contained in the product.  
MM/DD/YY format is product-specific for: sage\_atmos\_dyn, sage\_atmos\_comp,  
erbe\_erpMMDDYYYY format is product-specific for: LARC\_FIRE, LARC\_GTE.

Valid Values : See 311-CD-107-001

**Column: RANGEENDINGTIME****Description**

The last hour (and optionally minute, or minute and second) of the temporal coverage period being described for granule or collection.

Valid Values : See 311-CD-107-001

**Column: REVISIONDATE****Description**

Represents the date and possibly the time that this directory entry was created or the latest date and time of its modification or update.

**Column: SEARCHABLE****Description**

This holds the searchable key for the ECS attribute.

**Column: SENSORCHARACTERISTICDATATYPE****Description**

The datatype of the Instrument Characteristic/attribute defined by  
InstrumentCharacteristicName.

Valid Values : See 311-CD-107-001

**Column: SENSORCHARACTERISTICDESC****Description**

A description of the attribute defined by SensorCharacteristicName. (e.g. SensorCharacteristicName=SensorDevice, SensorCharacteristicDescription= Charge coupled device).

Valid Values : See 311-CD-107-001

**Column: SENSORCHARACTERISTICNAME****Description**

The name of the Sensor Characteristic/attribute. Sensor attributes defined using SensorCharacteristicName must be single-valued attributes of the object 'Sensor' and not attributes of undefined objects.

Valid Values : See 311-CD-107-001

**Column: SENSORCHARACTERISTICUNIT****Description**

The unit of the Sensor Characteristic (e.g. nanometers).

Valid Values : See 311-CD-107-001

**Column: SENSORDESCRIPTION****Description**

Description of a sensor.

Valid Values : See 311-CD-107-001

**Column: SENSORLONGNAME****Description**

The generic or long name description of a sensor. (e.g. Visible-Near Infrared, Human Visual, Human Auditory).

Valid Values : See 311-CD-107-001

**Column: SENSORSHORTNAME****Description**

A sensor is a defined sensory sub-component of an instrument. (e.g. InstrumentShortName=ASTER, NumberofSensors= 3, SensorShortName= SWIR, SensorShortName= TIR, SensorShortName= VNIR) In cases where the Instrument has a single Sensor or the Instrument and Sensor are synonymous then both attributes should be populated. (e.g. AVHRR). Sensors cannot exist without Instruments.

Valid Values : See 311-CD-107-001

**Column: SENSORTECHNIQUE****Description**

The sensor technique. (e.g. laser altimetry).

Valid Values : See 311-CD-107-001

**Column: SHAPE****Description**

TBS

**Column: SHORTNAME****Description**

This is the short name for the collection. This is an ECS attribute.

Valid Values : See 311-CD-107-001

**Column: SITEID****Description**

This is the site identifier. This uniquely identifies the Data Archiver Center.

**Column: SITEIDAGG****Description**

TBS

**Column: SITEIDREF****Description**

This is the site identifier. This uniquely identifies the Data Archiver Center reference.

**Column: SITENAME****Description**

This holds the site name.

**Column: SOUTHBOUNDINGCOORDINATE****Description**

Southern-most limit of coverage expressed in geodetic latitude.

Valid Values : See 311-CD-107-001

**Column: SPATIALKEYWORD****Description**

This attribute specifies a word or phrase which serves to summarize the spatial regions covered by the collection. It may be repeated if several regions are covered. This often occurs when a collection is described as covering some large region, and several smaller subregions within that region.

Valid Values : See 311-CD-107-001

**Column: STARTDATETIME****Description**

The starting date of a collection.

Valid Values : See 311-CD-107-001

**Column: STRINGSENSORCHARVALUE****Description**

This holds the string value of the sensor.

Valid Values : See 311-CD-107-001

**Column: STRINSTRUMENTCHARVALUE****Description**

This holds the string value of the instrument.

Valid Values : See 311-CD-107-001

**Column: STRPLATFORMCHARACTERISTICVALUE****Description**

This holds the string value of the platform.

Valid Values : See 311-CD-107-001

**Column: SUGGESTEDUSAGE****Description**

This attribute describes how this collection or granule may be best used to support earth science/global change research.

Valid Values : See 311-CD-107-001

**Column: TEMPORALKEYWORD****Description**

This attribute specifies a word or phrase which serves to summarize the temporal characteristics referenced in the collection. i.e. Monthly Composite, Annual Mean.

Valid Values : See 311-CD-107-001

**Column: TEMPORALRANGETYPE****Description**

This attribute tells the system and ultimately the end user how temporal coverage is specified for the collection, granule, or event.

Valid Values : See 311-CD-107-001

**Column: TERMDESCRIPTION****Description**

This contains the description for a term.

**Column: TIMETYPE****Description**

This attribute provides the time system which the values found in temporal subclasses represent.

**Column: TOPICDESCRIPTION****Description**

This is the topic description.

**Column: UNITS****Description**

TBS

**Column: VARIABLEDESCRIPTION****Description**

This is the variable description

**Column: VERSIONID****Description**

This is the version identifier for the collection.

Valid Values : See 311-CD-107-001

**Column: WESTBOUNDINGCOORDINATE****Description**

Western-most coordinate of the limit of coverage expressed in longitude.

Valid Values : See 311-CD-107-001

#### **4.1.4 Column Domains**

Domains specify the ranges of values allowed for a given table column. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types. There are no domains defined in the DM database.

#### **4.1.5 Rules**

Sybase supports the definitions of rules. Rules provide a means for enforcing domain constraints on a given column. All rules defined in Sybase for the DM database are described herein.

#### **4.1.6 Defaults**

Defaults are used to supply a value for a column when one is not defined at insert time. All defaults defined in Sybase in the DM database are described herein.

#### **4.1.7 Views**

Sybase allows the definition of views as a means of limiting an application or users access to data in a table or tables. Views create a logical table from columns found in one or more tables. All views defined in the DM databases are described herein.

##### **View List**

Name	Description
DmDdStringAttributesUv	To Be Supplied
DmDdTTemporalAttributesUv	To Be Supplied

##### **View: DmDdStringAttributesUv**

###### **Code**

```
create view DmDdStringAttributesUv
as
select
    A.collectionId collectionId,
    A.siteId siteId,
    A.AdditionalAttributeName AdditionalAttributeName,
    A.attributeLevelId attributeLevelId,
    B.keywordName keywordName
from
    DmDdAdditionalAttributes A, DmDdStringDomain B
where
    A.siteId = B.siteId
and
    A.attributeId = B.attributeId
and
    A.AdditionalAttributeDataType = "S"
```

##### **View: DmDdTTemporalAttributesUv**

###### **Code**

```
create view DmDdTTemporalAttributesUv
as
select
    A.collectionId collectionId,
    A.siteId siteId,
    A.AdditionalAttributeName AdditionalAttributeName,
    A.attributeLevelId attributeLevelId,
```

B.StartDateTime StartDateTime,  
B.EndDateTime EndDateTime  
from DmDdAdditionalAttributes A, DmDdDateTimeDomain B  
where A.siteId = B.siteId  
and A.attributeId = B.attributeId  
and A.AdditionalAttributeDataType = "D"

#### **4.1.8 Integrity Constraints**

Sybase version 11.0.1 allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks without automatically without requiring programming. Sybase 11 is only ANSI-92 compliant, however, therefore its constraints support “restrict-only” operations. This means that a row can not be deleted or updated if their are rows in other tables having a foreign key dependency on that row. Cascade delete and update operations can not be performed if a declarative constraint has been used. All declarative integrity constraints defined in the DM database are described herein.

#### **Dependencies on Table: DmDdCollECSAttributeXref**

##### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdAttributeXref	siteld attributeld	siteld attributeld

#### **Dependencies on Table: DmDdECSAttributes**

##### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdCollECSAttributeXref	ECSAttributeName	ECSAttributeName

#### **Dependencies on Table: DmDdECSCollection**

##### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdBoundingRectangle		
DmDdCollECSAttributeXref	collectionId siteld	collectionId siteld collectionId siteld

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdEquivalentAttributes	collectionId siteId	collectionId siteId
DmDdMultipleCollection	collectionId siteId	collectionId siteId
DmDdSingleTypeCollection	collectionId siteId	collectionId siteId
DmDdSpatialKeywordClass	collectionId siteId	collectionId siteId
DmDdTemporal	collectionId siteId	collectionId siteId
DmDdTemporalKeywordClass	collectionId siteId	collectionId siteId
DmDdAdditionalAttributes	collectionId siteId	collectionId siteId
DmDdInfoMgrCollXref	collectionId siteId	collectionId siteId
DmDdInstrumentXref	collectionId siteId	collectionId siteId
DmDdPlatformXref	collectionId siteId	collectionId siteId
DmDdSensorXref	collectionId siteId	collectionId siteId
DmDdECSCollKeywordXref	collectionId siteId	collectionId siteId

### **Dependencies on Table: DmDdECSCollKeywordXref**

#### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdKeywordParameterXref	keywordId siteId	keywordId siteId

## **Dependencies on Table: DmDdECSDiscipline**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSDisciplineKeyword	ECSDisciplineKeyword

## **Dependencies on Table: DmDdECSParameterDetails**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdKeywordParameterXref	ECSParameterKeyword	ECSParameterKeyword

## **Dependencies on Table: DmDdECSTerm**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSTermKeyword	ECSTermKeyword

## **Dependencies on Table: DmDdECSTopic**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSTopicKeyword	ECSTopicKeyword

## **Dependencies on Table: DmDdECSVariable**

### **Reference by List**

Referenced by	Primary Key	Foreign Key
DmDdECSCollKeywordXref	ECSVariableKeyword	ECSVariableKeyword

## Dependencies on Table: DmDdEquivalentAttributes

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdAttributeXref	sitId attributeId	sitIdRef attributeIdRef

## Dependencies on Table: DmDdGroup

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdGroupXref	groupName	groupName
DmDdGroupXref	groupName	groupParentName
DmDdLevelGroupXref	groupName	groupName

## Dependencies on Table: DmDdInfoMgr

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdInfoMgrCollXref	infoMgrName	infoMgrName
DmDdInfoMgrXref	infoMgrName	infoMgrName
DmDdInfoMgrXref	infoMgrName	infoMgrParentName

## Dependencies on Table: DmDdInstrument

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdInstrumentCharacteristic	InstrumentShortName	InstrumentShortName
DmDdInstrumentXref	InstrumentShortName	InstrumentShortName
DmDdOperationModeClass	InstrumentShortName	InstrumentShortName

Referenced by	Primary Key	Foreign Key
DmDdSensorInstrumentXref	InstrumentShortName	InstrumentShortName
DmDdInstrumentPlatformXref	InstrumentShortName	InstrumentShortName

### Dependencies on Table: DmDdInstrumentCharacteristic

#### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdDateInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdFloatInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdIntInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName
DmDdStringInstrumentChar	InstrumentCharacteristicName	InstrumentCharacteristicName

### Dependencies on Table: DmDdMetadataLevel

#### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdAdditionalAttributes	levelId	attributeLevelId
DmDdECSAttributes	levelId	attributeLevelId
DmDdLevelGroupXref	levelId	levelId

### Dependencies on Table: DmDdMultipleCollection

#### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdCollectionXref	collectionId siteId	collectionIdRef siteIdRef
DmDdCollectionXref	collectionId siteId	collectionIdAgg siteIdAgg

## **Dependencies on Table: DmDdPlatform**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdInstrumentPlatformXref	PlatformShortName	PlatformShortName
DmDdPlatformCharacteristic	PlatformShortName	PlatformShortName
DmDdPlatformXref	PlatformShortName	PlatformShortName

## **Dependencies on Table: DmDdPlatformCharacteristic**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdDatePlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdFloatPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdIntPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName
DmDdStringPlatformChar	PlatformCharacteristicName	PlatformCharacteristicName

## **Dependencies on Table: DmDdSensor**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdSensorCharacteristic	SensorShortName	SensorShortName
DmDdSensorXref	SensorShortName	SensorShortName
DmDdSensorInstrumentXref	SensorShortName	SensorShortName

## **Dependencies on Table: DmDdSensorCharacteristic**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdDateSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdFloatSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdIntSensorChar	SensorCharacteristicName	SensorCharacteristicName
DmDdStringSensorChar	SensorCharacteristicName	SensorCharacteristicName

## **Dependencies on Table: DmDdSingleTypeCollection**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdCollectionXref	collectionId siteId	collectionIdRef siteIdRef

## **Dependencies on Table: DmDdStringDomain**

### **Reference by List**

<b>Referenced by</b>	<b>Primary Key</b>	<b>Foreign Key</b>
DmDdKeywordAliasXref	siteId attributeId keywordName	siteId attributeId keywordName
DmDdKeywordAliasXref	siteId attributeId keywordName	siteIdRef attributeIdRef keywordNameRef

## Dependencies on Table: DmDdStringType

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdStringDomain	siteld attributeld	siteld attributeld

## Dependencies on Table: DmDdTTemporal

### Reference by List

Referenced by	Primary Key	Foreign Key
DmDdRangeDateTime	collectionId siteld	collectionId siteld

## Dependencies on Table: DmDdTTemporalKeywordClass

### Reference by List - TBS

#### 4.1.9 Triggers

Sybase supports the enforcement of business policy via the use of triggers. A trigger is best defined as set of activities or checks that should be performed automatically whenever a row is inserted, updated, or deleted from a given table. Sybase version 11.0.1 allows the definition of insert, update, and delete trigger per table. A listing of each the triggers in the DM database is given here. A brief definition of each of these triggers follows.

### Trigger List

Table	Trigger	Description
DmDdAdditionalAttributes	ti_dmddadditionalattributes	TBS
DmDdAdditionalAttributes	TrigDelAddiAttribute	TBS
DmDdAttributeXref	ti_dmddattributexref	TBS

<b>Table</b>	<b>Trigger</b>	<b>Description</b>
DmDdAttributeXref	TrigUpdDmDdAttributeXref	TBS
DmDdBoundingRectangle	ti_dmddboundingrectangle	TBS
DmDdCollECSAttributeXref	ti_dmddcollectsattributexref	TBS
DmDdCollECSAttributeXref	TrigDelCollECSARef	TBS
DmDdCollectionXref	ti_dmddcollectionxref	TBS
DmDdDateInstrumentChar	ti_dmdddateinstrumentchar	TBS
DmDdDatePlatformChar	ti_dmdddateplatformchar	TBS
DmDdDateSensorChar	ti_dmddatesensorchar	TBS
DmDdDateTimeDomain	TrigInsDmDdDateTimeDomain	TBS
DmDdDateTimeDomain	TrigUpdDateTimeDomain	TBS
DmDdECSAttributes	ti_dmddecsattributes	TBS
DmDdECSCollection	TrigUpdDmDdECSCollection	TBS
DmDdECSCollKeywordXref	ti_dmddecscollkeywordxref	TBS
DmDdEquivalentAttributes	ti_dmddequivalentattributes	TBS
DmDdEquivalentAttributes	TrigDelEquicAttribute	TBS
DmDdFloatInstrumentChar	ti_dmddfloatinstrumentchar	TBS
DmDdFloatPlatformChar	ti_dmddfloatplatformchar	TBS
DmDdFloatSensorChar	ti_dmddfloatsensorchar	TBS
DmDdGroupXref	ti_dmddgroupxref	TBS
DmDdInfoMgrCollXref	TrigInsUpdDmDdInfoMgrCollXref	TBS
DmDdInfoMgrCollXref	TrigInsUpdDmDdInfoMgrCollXref	TBS
DmDdInfoMgrXref	ti_dmddinfomgrxref	TBS
DmDdInstrumentCharacteristic	ti_dmddinstrumentcharacteristi	TBS
DmDdInstrumentPlatformXref	ti_dmddinstrumentplatformxref	TBS
DmDdInstrumentXref	ti_dmddinstrumentxref	TBS
DmDdIntInstrumentChar	ti_dmddintinstrumentchar	TBS
DmDdIntPlatformChar	ti_dmddintplatformchar	TBS
DmDdIntSensorChar	ti_dmddintsensorchar	TBS

<b>Table</b>	<b>Trigger</b>	<b>Description</b>
DmDdKeywordAliasXref	ti_dmddkeywordaliasxref	TBS
DmDdKeywordParameterXref	ti_dmddkeywordparameterxref	TBS
DmDdLevelGroupXref	ti_dmddlevelgroupxref	TBS
DmDdMultipleCollection	ti_dmddmultiplecollection	TBS
DmDdNumericDomain	TrigInsDmDdNumDomain	TBS
DmDdNumericDomain	TrigUpdDmDdNumericDomain	TBS
DmDdOperationModeClass	ti_dmddoperationmodeclass	TBS
DmDdPlatformCharacteristic	ti_dmddplatformcharacteristic	TBS
DmDdPlatformXref	ti_dmddplatformxref	TBS
DmDdRangeDateTime	ti_dmddrangedatetime	TBS
DmDdSensorCharacteristic	ti_dmddsensorcharacteristic	TBS
DmDdSensorInstrumentXref	ti_dmddsensorinstrumentxref	TBS
DmDdSensorXref	ti_dmddsensorxref	TBS
DmDdSingleTypeCollection	ti_dmddsingletypecollection	TBS
DmDdSpatialDomain	TrigInsDmDdSpatialDomain	TBS
DmDdSpatialDomain	TrigUpdDmDdSpatialDomain	TBS
DmDdSpatialKeywordClass	ti_dmddspatialkeywordclass	TBS
DmDdStringDomain	ti_dmddstringdomain	TBS
DmDdStringInstrumentChar	ti_dmddstringinstrumentchar	TBS
DmDdStringPlatformChar	ti_dmddstringplatformchar	TBS
DmDdStringSensorChar	ti_dmddstringsensorchar	TBS
DmDdStringType	TrigInsDmDdStringType	TBS
DmDdStringType	TrigUpdDmDdStringType	TBS
DmDdTemporal	ti_dmddtemporal	TBS
DmDdTemporalKeywordClass	ti_dmddtemporalkeywordclass	TBS

## **Trigger: ti\_dmddadditionalattributes**

### **Trigger Code**

```
/* Insert trigger "ti_dmddadditionalattributes" for table "DmDdAdditionalAttributes" */
create trigger ti_dmddadditionalattributes on DmDdAdditionalAttributes for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdECSCollection" must exist when inserting a child in
    "DmDdAdditionalAttributes" */
    if update(collectionId) or
        update(siteId)
    begin
        if (select count(*)
            from DmDdECSCollection t1, inserted t2
            where t1.collectionId = t2.collectionId
            and t1.siteId = t2.siteId) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
    "DmDdAdditionalAttributes".'
            goto error
        end
    end

    /* Parent "DmDdMetadataLevel" must exist when inserting a child in
    "DmDdAdditionalAttributes" */
    if update(attributeLevelId)
    begin
        select @numnull = (select count(*)
            from inserted
            where attributeLevelId is null)
        if @numnull != @numrows
            if (select count(*)
                from DmDdMetadataLevel t1, inserted t2
                where t1.levelId = t2.attributeLevelId) != @numrows - @numnull
            begin
                select @errno = 30002,
                    @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
    "DmDdAdditionalAttributes".'
                goto error
            end
    end
```

```

        end
    end

    return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: TrigDelAddiAttribute**

#### **Trigger Code**

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigDelAddiAttribute
--
--TABLE ACCESSED:    DmDdAdditionalAttributes
--                    DmDdStringType
--                    DmDdDateTimeDomain
--                    DmDdNumericDomain
--                    DmDdSpatialDomain
--                    DmDdAttributeXref
--                    deleted
--
--*****
CREATE TRIGGER TrigDelAddiAttribute
ON DmDdAdditionalAttributes
FOR DELETE
AS
BEGIN

    IF (SELECT count(*) FROM deleted d, DmDdStringType s
        WHERE d.siteId = s.siteId
        AND d.attributeId = s.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
        WHERE d.siteId = a.siteId
        AND d.attributeId = a.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
        WHERE d.siteId = n.siteId
        AND d.attributeId = n.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
        WHERE d.siteId = s.siteId
        AND d.attributeId = s.attributeId) > 0
    BEGIN

```

```

    INSERT DmDdAdditionalAttributes
    SELECT * FROM deleted

    RAISERROR 30515
    "The record in DmDdAdditionalAttributes is still referred by other data,
deleting has been aborted"
    RETURN
END
RETURN
go

```

### **Trigger: ti\_dmddattributexref**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddattributexref" for table "DmDdAttributeXref" */
create trigger ti_dmddattributexref on DmDdAttributeXref for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdEquivalentAttributes" must exist when inserting a child in
"DmDdAttributeXref" */
if update(siteIdRef) or
    update(attributeIdRef)
begin
if (select count(*)
    from DmDdEquivalentAttributes t1, inserted t2
    where t1.siteId = t2.siteIdRef
    and t1.attributeId = t2.attributeIdRef) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdEquivalentAttributes". Cannot create
child in "DmDdAttributeXref".'
        goto error
    end
end
end

/* Parent "DmDdCollECSAttributeXref" must exist when inserting a child in
"DmDdAttributeXref" */

```

```

if update(siteId) or
   update(attributeId)
begin
  if (select count(*)
      from DmDdCollECSAttributeXref t1, inserted t2
      where t1.siteId = t2.siteId
        and t1.attributeId = t2.attributeId) != @numrows
  begin
    select @errno = 30002,
           @errmsg = 'Parent does not exist in "DmDdCollECSAttributeXref". Cannot create
child in "DmDdAttributeXref".'
    goto error
  end
end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: TrigUpdDmDdAttributeXref

### Trigger Code

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigUpdDmDdAttributeXref
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                  DmDdAttributeXref
--                  inserted
--                  deleted
--
--*****

```

```

CREATE TRIGGER TrigUpdDmDdAttributeXref
ON DmDdAttributeXref
FOR UPDATE
AS
BEGIN
  DECLARE @attributeIdIns      attIdType,
          @siteIdIns       int,
          @attributeIdDel attIdType,
          @siteIdDel       int,
          @attributeIdRefIns attIdType,

```

```

        @siteIdRefIns      int,
        @attributeIdRefDel attIdType,
        @siteIdRefDel      int,
        @error              int

BEGIN
    DECLARE updAttributeXrefCS CURSOR FOR
        SELECT i.siteId, i.attributeId, d.siteId, d.attributeId,
               i.siteIdRef, i.attributeIdRef, d.siteIdRef, d.attributeIdRef
        FROM inserted i, deleted d
        WHERE i.siteId *= d.siteId
        AND i.attributeId *= d.attributeId

    OPEN updAttributeXrefCS

    FETCH updAttributeXrefCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel,
        @siteIdRefIns,
        @attributeIdRefIns,
        @siteIdRefDel,
        @attributeIdRefDel

    WHILE @@sqlstatus != 2
    BEGIN
        SELECT @error = 0
        IF @siteIdDel = null AND @attributeIdDel = null
            AND @siteIdRefDel = null AND @attributeIdRefDel = null
        BEGIN

            IF (select count(*) FROM DmDdCollECSAttributeXref
                WHERE siteId = @siteIdIns
                AND attributeId = @attributeIdIns) = 0
            AND (SELECT count(*) FROM DmDdEquivalentAttributes
                WHERE siteId = @siteIdIns
                AND attributeId = @attributeIdIns) = 0
            BEGIN
                SELECT @error = 1
            END

            IF @error = 0
            BEGIN
                IF (select count(*) FROM DmDdCollECSAttributeXref
                    WHERE siteId = @siteIdRefIns
                    AND attributeId = @attributeIdRefIns) = 0
                AND (SELECT count(*) FROM DmDdEquivalentAttributes
                    WHERE siteId = @siteIdRefIns
                    AND attributeId = @attributeIdRefIns) = 0
                BEGIN
                    SELECT @error = 1
                END
            END
        END
    END

```

```

        END
    END

    IF @error = 1
    BEGIN
        DELETE DmDdAttributeXref
        FROM DmDdAttributeXref s, inserted i
        WHERE s.siteId = i.siteId
        AND s.attributeId = i.attributeId

        INSERT DmDdAttributeXref
        SELECT * from deleted

        RAISERROR 30515
        "No reference data exists for updated data in DmDdAttributeXref, updating
has been aborted"
        RETURN
    END

    END /* if @siteIdDel = null and @attributeIdDel = null */

    FETCH updAttributeXrefCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel,
        @siteIdRefIns,
        @attributeIdRefIns,
        @siteIdRefDel,
        @attributeIdRefDel

        END /* WHILE */
    END
    END
    RETURN
    go

```

### **Trigger: ti\_dmddboundingrectangle**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddboundingrectangle" for table "DmDdBoundingRectangle" */
create trigger ti_dmddboundingrectangle on DmDdBoundingRectangle for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

```

```

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdBoundingRectangle" */
if update(collectionId) or
    update(siteId)
begin
    if (select count(*)
        from DmDdECSCollection t1, inserted t2
        where t1.collectionId = t2.collectionId
        and t1.siteId = t2.siteId) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdBoundingRectangle".'
        goto error
    end
end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: ti\_dmddcollecsattributexref**

### **Trigger Code**

```

/* Insert trigger "ti_dmddcollecsattributexref" for table "DmDdCollECSAttributeXref" */
create trigger ti_dmddcollecsattributexref on DmDdCollECSAttributeXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

```

```

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdCollECSAttributeXref" */
if update(collectionId) or
    update(siteId)
begin
if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
    and t1.siteId = t2.siteId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdCollECSAttributeXref".'
        goto error
    end
end

/* Parent "DmDdECSAttributes" must exist when inserting a child in
"DmDdCollECSAttributeXref" */
if update(ECSAttributeName)
begin
if (select count(*)
    from DmDdECSAttributes t1, inserted t2
    where t1.ECSAttributeName = t2.ECSAttributeName) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSAttributes". Cannot create child in
"DmDdCollECSAttributeXref".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: TrigDelCollECSAXref

### Trigger Code

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigDelCollECSAttriXref
--
```

```

--TABLE ACCESSED:      DmDdCollECSAttributeXref
--                      DmDdStringType
--                      DmDdDateTimeDomain
--                      DmDdNumericDomain
--                      DmDdSpatialDomain
--                      DmDdAttributeXref
--                      deleted
--
--*****=====
CREATE TRIGGER TrigDelCollECSAXref
ON DmDdCollECSAttributeXref
FOR DELETE
AS
BEGIN

IF (SELECT count(*) FROM deleted d, DmDdStringType s
    WHERE d.siteId = s.siteId
    AND d.attributeId = s.attributeId) > 0
OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
    WHERE d.siteId = a.siteId
    AND d.attributeId = a.attributeId) > 0
OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
    WHERE d.siteId = n.siteId
    AND d.attributeId = n.attributeId) > 0
OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
    WHERE d.siteId = s.siteId
    AND d.attributeId = s.attributeId) > 0
OR (SELECT count(*) FROM deleted d, DmDdAttributeXref s
    WHERE d.siteId = s.siteId
    AND d.attributeId = s.attributeId
    OR d.siteId = s.siteIdRef
    AND d.attributeId = s.attributeIdRef) > 0
BEGIN

    INSERT DmDdCollECSAttributeXref
    SELECT * FROM deleted

    RAISERROR 30515
    "The record in DmDdCollECSAttributeXref is stil refered by other data,
deleting has been aborted"
    RETURN
END
END
RETURN
go

```

## **Trigger: ti\_dmddcollectionxref**

### **Trigger Code**

```
/* Insert trigger "ti_dmddcollectionxref" for table "DmDdCollectionXref" */
create trigger ti_dmddcollectionxref on DmDdCollectionXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdMultipleCollection" must exist when inserting a child in
    "DmDdCollectionXref" */
    if update(collectionIdRef) or
        update(siteIdRef)
    begin
        if (select count(*)
            from DmDdMultipleCollection t1, inserted t2
            where t1.collectionId = t2.collectionIdRef
            and t1.siteId = t2.siteIdRef) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdMultipleCollection". Cannot create child
in "DmDdCollectionXref".'
            goto error
        end
    end

    /* Parent "DmDdSingleTypeCollection" must exist when inserting a child in
    "DmDdCollectionXref" */
    if update(collectionIdRef) or
        update(siteIdRef)
    begin
        if (select count(*)
            from DmDdSingleTypeCollection t1, inserted t2
            where t1.collectionId = t2.collectionIdRef
            and t1.siteId = t2.siteIdRef) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdSingleTypeCollection". Cannot create
child in "DmDdCollectionXref".'
            goto error
        end
    end
```

```

/* Parent "DmDdMultipleCollection" must exist when inserting a child in
"DmDdCollectionXref" */
if update(collectionIdAgg) or
  update(siteIdAgg)
begin
  if (select count(*)
    from DmDdMultipleCollection t1, inserted t2
    where t1.collectionId = t2.collectionIdAgg
      and t1.siteId = t2.siteIdAgg) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdMultipleCollection". Cannot create child
in "DmDdCollectionXref".'
    goto error
  end
end
return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

### **Trigger: ti\_dmddateinstrumentchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddateinstrumentchar" for table "DmDdDateInstrumentChar" */
create trigger ti_dmddateinstrumentchar on DmDdDateInstrumentChar for insert as
begin
declare
  @numrows int,
  @numnull int,
  @errno int,
  @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
  return

/* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
"DmDdDateInstrumentChar" */
if update(InstrumentCharacteristicName)
begin
  if (select count(*)

```

```

from DmDdInstrumentCharacteristic t1, inserted t2
where t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create
child in "DmDdDateInstrumentChar".'
    goto error
end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddateplatformchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddateplatformchar" for table "DmDdDatePlatformChar" */
create trigger ti_dmddateplatformchar on DmDdDatePlatformChar for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdPlatformCharacteristic" must exist when inserting a child in
"DmDdDatePlatformChar" */
if update(PlatformCharacteristicName)
begin
    if (select count(*)
        from DmDdPlatformCharacteristic t1, inserted t2
        where t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create
child in "DmDdDatePlatformChar".'
        goto error
    end
end

```

```

        end
    end

    return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddatesensorchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddatesensorchar" for table "DmDdDateSensorChar" */
create trigger ti_dmddatesensorchar on DmDdDateSensorChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdSensorCharacteristic" must exist when inserting a child in
    "DmDdDateSensorChar" */
    if update(SensorCharacteristicName)
        begin
            if (select count(*)
                from DmDdSensorCharacteristic t1, inserted t2
                where t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
                begin
                    select @errno = 30002,
                        @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create
child in "DmDdDateSensorChar".'
                    goto error
                end
        end
    end

    return

/* Errors handling */
error:
    raiserror @errno @errmsg

```

```
rollback transaction  
end  
go
```

## Trigger: TrigInsDmDdDateTimeDomain

### Trigger Code

```
--*****  
--BEGIN PROLOG  
--  
--TRIGGER NAME:      TrigInsDmDdDateTimeDomain  
--  
--TABLE ACCESSED:    DmDdCollECSAttributeXref  
--                      DmDdAdditionalAttributes  
--                      DmDdDateTimeDomain  
--                      inserted  
--  
--*****  
  
CREATE TRIGGER TrigInsDmDdDateTimeDomain  
ON DmDdDateTimeDomain  
FOR INSERT  
AS  
BEGIN  
    DECLARE @attributeId      attIdType,  
            @siteId          int,  
            @hasIt           int  
  
    BEGIN  
        DECLARE insDateTimeDomainCS CURSOR FOR  
            SELECT siteId, attributeId  
            FROM inserted  
  
        OPEN insDateTimeDomainCS  
  
        FETCH insDateTimeDomainCS INTO  
            @siteId,  
            @attributeId  
  
        WHILE @@sqlstatus != 2  
        BEGIN  
            EXEC ProcCheckAttriExist @siteId = @siteId,  
                                @attributeId = @attributeId,  
                                @hasIt = @hasIt output  
            IF @hasIt = 0  
            BEGIN  
                DELETE DmDdDateTimeDomain  
                FROM DmDdDateTimeDomain s, inserted i  
                WHERE s.siteId = i.siteId  
            END  
        END  
    END  
END
```

```

        AND s.attributeId = i.attributeId

        RAISERROR 30515
        "No reference data exists for new record in DmDdDateTimeDomain,
inserting has been aborted"
        RETURN
    END

    FETCH insDateTimeDomainCS INTO
        @siteId,
        @attributeId
    END /* WHILE */
END
END
RETURN
go

```

## **Trigger: TrigUpdDateTimeDomain**

### **Trigger Code**

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigUpdDmDdDateTimeDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                  DmDdDateTimeDomain
--                  inserted
--                  deleted
--
--*****

```

```

CREATE TRIGGER TrigUpdDateTimeDomain
ON DmDdDateTimeDomain
FOR UPDATE
AS
BEGIN
    DECLARE @attributeIdIns      attIdType,
            @siteIdIns        int,
            @attributeIdDel   attIdType,
            @siteIdDel        int

    BEGIN
        DECLARE updDateTimeDomainCS CURSOR FOR
            SELECT i.siteId, i.attributeId, d.siteId, d.attributeId
            FROM inserted i, deleted d
            WHERE i.siteId *= d.siteId
            AND i.attributeId *= d.attributeId
    
```

```

OPEN updDateDomainCS

FETCH updDateDomainCS INTO
    @siteIdIns,
    @attributeIdIns,
    @siteIdDel,
    @attributeIdDel

WHILE @@sqlstatus != 2
BEGIN
    IF @siteIdDel = null AND @attributeIdDel = null
    BEGIN

        IF (select count(*) FROM DmDdCollECSAttributeXref
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdAdditionalAttributes
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdEquivalentAttributes
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        BEGIN
            DELETE DmDdDateTimeDomain
            FROM DmDdDateTimeDomain s, inserted i
            WHERE s.siteId = i.siteId
            AND s.attributeId = i.attributeId

            INSERT DmDdDateTimeDomain
            SELECT * from deleted

            RAISERROR 30515
            "No reference data exists for updated data in DmDdDateTimeDomain,
            updating has been aborted"
            RETURN
        END
    END /* if @siteIdDel = null and @attributeIdDel = null */

    FETCH updDateDomainCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel

    END /* WHILE */
END
END
RETURN
go

```

## **Trigger: ti\_dmddecsattributes**

### **Trigger Code**

```
/* Insert trigger "ti_dmddecsattributes" for table "DmDdECSAttributes" */
create trigger ti_dmddecsattributes on DmDdECSAttributes for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdMetadataLevel" must exist when inserting a child in "DmDdECSAttributes"
    */
    if update(attributeLevelId)
    begin
        select @numnull = (select count(*)
                            from inserted
                            where attributeLevelId is null)
        if @numnull != @numrows
            if (select count(*)
                from DmDdMetadataLevel t1, inserted t2
                where t1.levelId = t2.attributeLevelId) != @numrows - @numnull
                begin
                    select @errno = 30002,
                           @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
"DmDdECSAttributes".'
                    goto error
                end
            end
        end

        return

    /* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go
```

## **Trigger: TrigUpdDmDdECSCollection**

### **Trigger Code**

```
--|||||||||||||||||||||||||||||||||||||||||||
```

```

-- Name: TrigUpdDmDdECSCollection
--
-- Description:
--
-- Trigger that check duplicate ShortName
--
--////////////////////////////////////////////////////////////////////////

create trigger TrigUpdDmDdECSCollection on DmDdECSCollection
for update as
begin

    if @@rowcount = 0
        return

    declare shortName_crsr cursor for
        select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
        from inserted c, DmDdInfoMgrCollXref i, DmDdInfoMgr m
        where i.collectionId = c.collectionId
        and i.siteId = c.siteId
        and i.infoMgrName = m.infoMgrName
        and m.infoMgrType = "SDSRV"

    open shortName_crsr

    declare
        @infoMgrName      char(20),
        @collectionId int,
        @siteId           int,
        @shortName        char(20)

    fetch shortName_crsr into
        @infoMgrName,
        @collectionId,
        @siteId,
        @shortName

    while @@sqlstatus != 2
    begin

        if(select count(*)
            from DmDdInfoMgrCollXref i, DmDdECSCollection c
            where i.collectionId = c.collectionId
            and i.siteId = c.siteId
            and i.infoMgrName = @infoMgrName
            and c.ShortName = @shortName
            and (i.collectionId != @collectionId or i.siteId != @siteId)
            ) > 0
        begin

            raiserror 25100 "Duplicate SortName found, update failed"
            rollback transaction
        end
    end
end

```

```

        return
end -- if(select count(*)

fetch shortName_crsr into
@infoMgrName,
@collectionId,
@siteId,
@shortName

end --while
close shortName_crsr
end
return
go

```

### **Trigger: ti\_dmddecscollkeywordxref**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddecscollkeywordxref" for table "DmDdECSCollKeywordXref" */
create trigger ti_dmddecscollkeywordxref on DmDdECSCollKeywordXref for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdECSCollKeywordXref" */
if update(collectionId) or
    update(siteId)
begin
if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
    and t1.siteId = t2.siteId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdECSCollKeywordXref".'
    goto error
end
end
end

```

```

/* Parent "DmDdECSDiscipline" must exist when inserting a child in
"DmDdECSCollKeywordXref" */
if update(ECSDisciplineKeyword)
begin
    select @numnull = (select count(*)
        from inserted
        where ECSDisciplineKeyword is null)
    if @numnull != @numrows
        if (select count(*)
            from DmDdECSDiscipline t1, inserted t2
            where t1.ECSDisciplineKeyword = t2.ECSDisciplineKeyword) != @numrows -
@numnull
            begin
                select @errno = 30002,
                    @errmsg = 'Parent does not exist in "DmDdECSDiscipline". Cannot create child in
"DmDdECSCollKeywordXref".'
                goto error
            end
        end
    end

/* Parent "DmDdECSTopic" must exist when inserting a child in
"DmDdECSCollKeywordXref" */
if update(ECSTopicKeyword)
begin
    select @numnull = (select count(*)
        from inserted
        where ECSTopicKeyword is null)
    if @numnull != @numrows
        if (select count(*)
            from DmDdECSTopic t1, inserted t2
            where t1.ECSTopicKeyword = t2.ECSTopicKeyword) != @numrows - @numnull
            begin
                select @errno = 30002,
                    @errmsg = 'Parent does not exist in "DmDdECSTopic". Cannot create child in
"DmDdECSCollKeywordXref".'
                goto error
            end
        end
    end

/* Parent "DmDdECSVariable" must exist when inserting a child in
"DmDdECSCollKeywordXref" */
if update(ECSVariableKeyword)
begin
    select @numnull = (select count(*)
        from inserted
        where ECSVariableKeyword is null)
    if @numnull != @numrows
        if (select count(*)
            from DmDdECSVariable t1, inserted t2
            where t1.ECSVariableKeyword = t2.ECSVariableKeyword) != @numrows -
@numnull
            begin

```

```

        select @errno = 30002,
               @errmsg = 'Parent does not exist in "DmDdECSVariable". Cannot create child in
"DmDdECSCollKeywordXref".'
        goto error
      end
    end

/* Parent "DmDdECSTerm" must exist when inserting a child in
*DmDdECSCollKeywordXref" */
if update(ECSTermKeyword)
begin
  select @numnull = (select count(*)
                     from inserted
                     where ECSTermKeyword is null)
  if @numnull != @numrows
    if (select count(*)
        from DmDdECSTerm t1, inserted t2
        where t1.ECSTermKeyword = t2.ECSTermKeyword) != @numrows - @numnull
  begin
    select @errno = 30002,
           @errmsg = 'Parent does not exist in "DmDdECSTerm". Cannot create child in
*DmDdECSCollKeywordXref".'
    goto error
  end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## **Trigger: ti\_dmddequivalentattributes**

### **Trigger Code**

```

/* Insert trigger "ti_dmddequivalentattributes" for table "DmDdEquivalentAttributes" */
create trigger ti_dmddequivalentattributes on DmDdEquivalentAttributes for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0

```

```

    return

    /* Parent "DmDdECSCollection" must exist when inserting a child in
    "DmDdEquivalentAttributes" */
    if update(collectionId) or
        update(siteId)
    begin
        if (select count(*)
            from DmDdECSCollection t1, inserted t2
            where t1.collectionId = t2.collectionId
            and t1.siteId = t2.siteId) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
    "DmDdEquivalentAttributes".'
            goto error
        end
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: TrigDelEquicAttribute**

### **Trigger Code**

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigDelEquicAttribute
--
--TABLE ACCESSED:   DmDdAdditionalAttributes
--                  DmDdStringType
--                  DmDdDateTimeDomain
--                  DmDdNumericDomain
--                  DmDdSpatialDomain
--                  DmDdAttributeXref
--                  deleted
--
--*****

```

```

CREATE TRIGGER TrigDelEquicAttribute
ON DmDdEquivalentAttributes

```

```

FOR DELETE
AS
BEGIN

    IF (SELECT count(*) FROM deleted d, DmDdStringType s
        WHERE d.siteId = s.siteId
        AND d.attributeId = s.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdDateTimeDomain a
        WHERE d.siteId = a.siteId
        AND d.attributeId = a.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdNumericDomain n
        WHERE d.siteId = n.siteId
        AND d.attributeId = n.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdSpatialDomain s
        WHERE d.siteId = s.siteId
        AND d.attributeId = s.attributeId) > 0
    OR (SELECT count(*) FROM deleted d, DmDdAttributeXref s
        WHERE d.siteId = s.siteId
        AND d.attributeId = s.attributeId
        OR d.siteId = s.siteIdRef
        AND d.attributeId = s.attributeIdRef) > 0
    BEGIN

        INSERT DmDdEquivalentAttributes
        SELECT * FROM deleted

        RAISERROR 30515
        "The record in DmDdEquivalentAttributes is still referred by other data,
deleting has been aborted"
        RETURN
    END
END
RETURN
go

```

### **Trigger: ti\_dmddfloatinstrumentchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddfloatinstrumentchar" for table "DmDdFloatInstrumentChar" */
create trigger ti_dmddfloatinstrumentchar on DmDdFloatInstrumentChar for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0

```

```

    return

    /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
    "DmDdFloatInstrumentChar" */
    if update(InstrumentCharacteristicName)
    begin
        if (select count(*)
            from DmDdInstrumentCharacteristic t1, inserted t2
            where t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) !=

@numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create
child in "DmDdFloatInstrumentChar".'
            goto error
        end
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddfloatplatformchar

### Trigger Code

```

/* Insert trigger "ti_dmddfloatplatformchar" for table "DmDdFloatPlatformChar" */
create trigger ti_dmddfloatplatformchar on DmDdFloatPlatformChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child in
    "DmDdFloatPlatformChar" */
    if update(PlatformCharacteristicName)
    begin

```

```

if (select count(*)
    from DmDdPlatformCharacteristic t1, inserted t2
    where t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create
child in "DmDdFloatPlatformChar".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddfloatsensorchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddfloatsensorchar" for table "DmDdFloatSensorChar" */
create trigger ti_dmddfloatsensorchar on DmDdFloatSensorChar for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdSensorCharacteristic" must exist when inserting a child in
"DmDdFloatSensorChar" */
if update(SensorCharacteristicName)
begin
    if (select count(*)
        from DmDdSensorCharacteristic t1, inserted t2
        where t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create
child in "DmDdFloatSensorChar".'
            goto error
    end
end

```

```

        end
    end

    return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddgroupxref**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddgroupxref" for table "DmDdGroupXref" */
create trigger ti_dmddgroupxref on DmDdGroupXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdGroup" must exist when inserting a child in "DmDdGroupXref" */
    if update(groupName)
        begin
            if (select count(*)
                from DmDdGroup t1, inserted t2
                where t1.groupName = t2.groupName) != @numrows
                begin
                    select @errno = 30002,
                        @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdGroupXref".'
                    goto error
                end
        end
    end

    /* Parent "DmDdGroup" must exist when inserting a child in "DmDdGroupXref" */
    if update(groupParentName)
        begin
            if (select count(*)
                from DmDdGroup t1, inserted t2
                where t1.groupName = t2.groupParentName) != @numrows

```

```

begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdGroupXref".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: TrigInsUpdDmDdInfoMgrCollXref**

### **Trigger Code**

```

--///////////////////////////////
-- Name: TrigInsUpdDmDdInfoMgrCollXref
--
-- Description:
--
-- Trigger that check duplicate ShortName
--
--/////////////////////////////

create trigger TrigInsUpdDmDdInfoMgrCollXref on DmDdInfoMgrCollXref
for insert, update as
begin

if @@rowcount = 0
    return

declare shortName_csr cursor for
    select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
    from inserted i, DmDdECSCollection c, DmDdInfoMgr m
    where i.collectionId = c.collectionId
    and i.siteId = c.siteId
    and i.infoMgrName = m.infoMgrName
    and m.infoMgrType = "SDSRV"

open shortName_csr

declare
    @infoMgrName      char(20),
    @collectionId int,

```

```

@siteId           int,
@shortName       char(20)

fetch shortName_crsr into
    @infoMgrName,
    @collectionId,
    @siteId,
    @shortName

while @@sqlstatus != 2
begin

if(select count(*)
    from DmDdInfoMgrCollXref i, DmDdECSCollection c
    where i.collectionId = c.collectionId
    and i.siteId = c.siteId
    and i.infoMgrName = @infoMgrName
    and c.ShortName = @shortName
    and (i.collectionId != @collectionId or i.siteId != @siteId)
) > 0
begin

    if (select count(*) from deleted) > 0 --update
    begin
        raiserror 25100 "Duplicate SortName found, update failed"
        rollback transaction
        return
    end
    else
    begin
        raiserror 25100 "Duplicate SortName found, inserting records in
DmDdInfoMgrCollXref and DmDdECSCollection have been deleted"
        declare delColl_crsr cursor for
            select collectionId, siteId
            from inserted
        open delColl_crsr
        while @@sqlstatus != 2
        begin
            fetch delColl_crsr into
                @collectionId,
                @siteId

                delete DmDdInfoMgrCollXref
                where collectionId = @collectionId
                and siteId = @siteId

                delete DmDdECSCollection
                where collectionId = @collectionId
                and siteId = @siteId

            fetch delColl_crsr into
                @collectionId,

```

```

        @siteId
    end
    close delColl_crsr
    return
end --if (select count(*) from deleted) > 0
end -- if(select count(*))

fetch shortName_crsr into
@infoMgrName,
@collectionId,
@siteId,
@shortName

end --while
close shortName_crsr
end
return
go

```

### **Trigger: TrigInsUpdDmDdInfoMgrCollXref**

#### **Trigger Code**

```

--///////////////////////////////
-- Name: TrigInsUpdDmDdInfoMgrCollXref
--
-- Description:
--
-- Trigger that check duplicate ShortName
--
--///////////////////////////////

create trigger TrigInsUpdDmDdInfoMgrCollXref on DmDdInfoMgrCollXref
for insert, update as
begin

    if @@rowcount = 0
        return

    declare shortName_crsr cursor for
        select i.infoMgrName, i.collectionId, i.siteId, c.ShortName
        from inserted i, DmDdECSCollection c, DmDdInfoMgr m
        where i.collectionId = c.collectionId
        and i.siteId = c.siteId
        and i.infoMgrName = m.infoMgrName
        and m.infoMgrType = "SDSRV"

    open shortName_crsr

    declare

```

```

@infoMgrName      char(20),
@collectionId int,
@siteId           int,
@shortName        char(20)

fetch shortName_csr into
    @infoMgrName,
    @collectionId,
    @siteId,
    @shortName

while @@sqlstatus != 2
begin

if(select count(*)
    from DmDdInfoMgrCollXref i, DmDdECSCollection c
    where i.collectionId = c.collectionId
    and i.siteId = c.siteId
    and i.infoMgrName = @infoMgrName
    and c.ShortName = @shortName
    and (i.collectionId != @collectionId or i.siteId != @siteId)
) > 0
begin

if (select count(*) from deleted) > 0 --update
begin
    raiserror 25100 "Duplicate SortName found, update failed"
    rollback transaction
    return
end
else
begin
    raiserror 25100 "Duplicate SortName found, inserting records in
DmDdInfoMgrCollXref and DmDdECSCollection have been deleted"
    declare delColl_csr cursor for
        select collectionId, siteId
        from inserted
    open delColl_csr
    while @@sqlstatus != 2
    begin
        fetch delColl_csr into
            @collectionId,
            @siteId

        delete DmDdInfoMgrCollXref
        where collectionId = @collectionId
        and siteId = @siteId

        delete DmDdECSCollection
        where collectionId = @collectionId
        and siteId = @siteId
    end
end
end
end

```

```

        fetch delColl_crsr into
        @collectionId,
        @siteId
    end
    close delColl_crsr
    return
end --if (select count(*) from deleted) > 0
end -- if(select count(*))

fetch shortName_crsr into
@infoMgrName,
@collectionId,
@siteId,
@shortName

end --while
close shortName_crsr
end
return
go

```

### **Trigger: ti\_dmddinfomgrxref**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddinfomgrxref" for table "DmDdInfoMgrXref" */
create trigger ti_dmddinfomgrxref on DmDdInfoMgrXref for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdInfoMgr" must exist when inserting a child in "DmDdInfoMgrXref" */
if update(infoMgrName)
begin
    if (select count(*)
        from DmDdInfoMgr t1, inserted t2
        where t1.infoMgrName = t2.infoMgrName) != @numrows
    begin
        select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdInfoMgr". Cannot create child in
"DmDdInfoMgrXref".'
        goto error
    end
end

```

```

        end
    end

/* Parent "DmDdInfoMgr" must exist when inserting a child in "DmDdInfoMgrXref" */
if update(infoMgrParentName)
begin
    if (select count(*)
        from DmDdInfoMgr t1, inserted t2
        where t1.infoMgrName = t2.infoMgrParentName) != @numrows
    begin
        select @errno = 30002,
              @errmsg = 'Parent does not exist in "DmDdInfoMgr". Cannot create child in
"DmDdInfoMgrXref".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddinstrumentcharacteristi

### Trigger Code

```

/* Insert trigger "ti_dmddinstrumentcharacteristi" for table "DmDdInstrumentCharacteristic" */
create trigger ti_dmddinstrumentcharacteristi on DmDdInstrumentCharacteristic for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdInstrument" must exist when inserting a child in
    "DmDdInstrumentCharacteristic" */
    if update(InstrumentShortName)
    begin
        select @numnull = (select count(*)
                           from inserted

```

```

        where InstrumentShortName is null)
if @numnull != @numrows
  if (select count(*)
      from DmDdInstrument t1, inserted t2
      where t1.InstrumentShortName = t2.InstrumentShortName) != @numrows - @numnull
    begin
      select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentCharacteristic".'
      goto error
    end
  end
return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: ti\_dmddinstrumentplatformxref

### Trigger Code

```

/* Insert trigger "ti_dmddinstrumentplatformxref" for table "DmDdInstrumentPlatformXref" */
create trigger ti_dmddinstrumentplatformxref on DmDdInstrumentPlatformXref for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

  /* Parent "DmDdPlatform" must exist when inserting a child in
  "DmDdInstrumentPlatformXref" */
  if update(PlatformShortName)
    begin
      if (select count(*)
          from DmDdPlatform t1, inserted t2
          where t1.PlatformShortName = t2.PlatformShortName) != @numrows
      begin
        select @errno = 30002,
              @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
              "DmDdInstrumentPlatformXref".'
        goto error
      end
    end
  end

```

```

"DmDdInstrumentPlatformXref".'
      goto error
    end
  end

/* Parent "DmDdInstrument" must exist when inserting a child in
"DmDdInstrumentPlatformXref" */
  if update(InstrumentShortName)
  begin
    if (select count(*)
        from DmDdInstrument t1, inserted t2
        where t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
    begin
      select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentPlatformXref".'
      goto error
    end
  end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: ti\_dmddinstrumentxref

### Trigger Code

```

/* Insert trigger "ti_dmddinstrumentxref" for table "DmDdInstrumentXref" */
create trigger ti_dmddinstrumentxref on DmDdInstrumentXref for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

/* Parent "DmDdInstrument" must exist when inserting a child in "DmDdInstrumentXref" */
  if update(InstrumentShortName)
  begin

```

```

if (select count(*)
    from DmDdInstrument t1, inserted t2
    where t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdInstrumentXref".'
        goto error
    end
end

/* Parent "DmDdECSCollection" must exist when inserting a child in
*DmDdInstrumentXref" */
if update(collectionId) or
    update(siteId)
begin
    if (select count(*)
        from DmDdECSCollection t1, inserted t2
        where t1.collectionId = t2.collectionId
        and t1.siteId = t2.siteId) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdInstrumentXref".'
            goto error
        end
    end
return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddintinstrumentchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddintinstrumentchar" for table "DmDdIntInstrumentChar" */
create trigger ti_dmddintinstrumentchar on DmDdIntInstrumentChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

```

```

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
"DmDdIntInstrumentChar" */
if update(InstrumentCharacteristicName)
begin
    if (select count(*)
        from DmDdInstrumentCharacteristic t1, inserted t2
        where t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create
child in "DmDdIntInstrumentChar".'
            goto error
        end
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: ti\_dmddintplatformchar**

### **Trigger Code**

```

/* Insert trigger "ti_dmddintplatformchar" for table "DmDdIntPlatformChar" */
create trigger ti_dmddintplatformchar on DmDdIntPlatformChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child in
    "DmDdIntPlatformChar" */

```

```

if update(PlatformCharacteristicName)
begin
  if (select count(*)
    from DmDdPlatformCharacteristic t1, inserted t2
    where t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create
child in "DmDdIntPlatformChar".'
    goto error
  end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: ti\_dmddintsensorchar

### Trigger Code

```

/* Insert trigger "ti_dmddintsensorchar" for table "DmDdIntSensorChar" */
create trigger ti_dmddintsensorchar on DmDdIntSensorChar for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

  /* Parent "DmDdSensorCharacteristic" must exist when inserting a child in
  "DmDdIntSensorChar" */
  if update(SensorCharacteristicName)
  begin
    if (select count(*)
      from DmDdSensorCharacteristic t1, inserted t2
      where t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows
    begin
      select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create

```

```

child in "DmDdIntSensorChar".'
    goto error
end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddkeywordaliasxref

### Trigger Code

```

/* Insert trigger "ti_dmddkeywordaliasxref" for table "DmDdKeywordAliasXref" */
create trigger ti_dmddkeywordaliasxref on DmDdKeywordAliasXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdStringDomain" must exist when inserting a child in
    "DmDdKeywordAliasXref" */
    if update(siteId) or
        update(attributeId) or
        update(keywordName)
    begin
        if (select count(*)
            from DmDdStringDomain t1, inserted t2
            where t1.siteId = t2.siteId
            and t1.attributeId = t2.attributeId
            and t1.keywordName = t2.keywordName) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdStringDomain". Cannot create child in
    "DmDdKeywordAliasXref".'
            goto error
        end
    end
end

```

```

/* Parent "DmDdStringDomain" must exist when inserting a child in
"DmDdKeywordAliasXref" */
if update(siteIdRef) or
  update(attributeIdRef) or
  update(keywordNameRef)
begin
  if (select count(*)
    from DmDdStringDomain t1, inserted t2
    where t1.siteId = t2.siteIdRef
    and t1.attributeId = t2.attributeIdRef
    and t1.keywordName = t2.keywordNameRef) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdStringDomain". Cannot create child in
"DmDdKeywordAliasXref".'
    goto error
  end
end
return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: ti\_dmddkeywordparameterxref

### Trigger Code

```

/* Insert trigger "ti_dmddkeywordparameterxref" for table "DmDdKeywordParameterXref" */
create trigger ti_dmddkeywordparameterxref on DmDdKeywordParameterXref for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

  /* Parent "DmDdECSCollKeywordXref" must exist when inserting a child in
  "DmDdKeywordParameterXref" */
  if update(keywordId) or

```

```

        update(siteId)
begin
if (select count(*)
    from DmDdECSCollKeywordXref t1, inserted t2
    where t1.keywordId = t2.keywordId
    and t1.siteId = t2.siteId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSCollKeywordXref". Cannot create
child in "DmDdKeywordParameterXref".'
        goto error
    end
end

/* Parent "DmDdECSPparameterDetails" must exist when inserting a child in
"DmDdKeywordParameterXref" */
if update(ECSPparameterKeyword)
begin
    if (select count(*)
        from DmDdECSPparameterDetails t1, inserted t2
        where t1.ECSPparameterKeyword = t2.ECSPparameterKeyword) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdECSPparameterDetails". Cannot create
child in "DmDdKeywordParameterXref".'
            goto error
        end
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: ti\_dmddlevelgroupxref**

### **Trigger Code**

```

/* Insert trigger "ti_dmddlevelgroupxref" for table "DmDdLevelGroupXref" */
create trigger ti_dmddlevelgroupxref on DmDdLevelGroupXref for insert as
begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

```

```

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdMetadataLevel" must exist when inserting a child in
"DmDdLevelGroupXref" */
if update(levelId)
begin
    if (select count(*)
        from DmDdMetadataLevel t1, inserted t2
        where t1.levelId = t2.levelId) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdMetadataLevel". Cannot create child in
"DmDdLevelGroupXref".'
        goto error
    end
end

/* Parent "DmDdGroup" must exist when inserting a child in "DmDdLevelGroupXref" */
if update(groupName)
begin
    if (select count(*)
        from DmDdGroup t1, inserted t2
        where t1.groupName = t2.groupName) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdGroup". Cannot create child in
"DmDdLevelGroupXref".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: ti\_dmddmultiplecollection**

### **Trigger Code**

```

/* Insert trigger "ti_dmddmultiplecollection" for table "DmDdMultipleCollection" */
create trigger ti_dmddmultiplecollection on DmDdMultipleCollection for insert as

```

```

begin
declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
    return

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdMultipleCollection" */
if update(collectionId) or
    update(siteId)
begin
if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
        and t1.siteId = t2.siteId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdMultipleCollection".'
    goto error
end
end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: TrigInsDmDdNumDomain

### Trigger Code

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigInsDmDdNumericDomain
--
--TABLE ACCESSED:    DmDdCollECSAttributeXref
--                      DmDdAdditionalAttributes
--                      DmDdNumericDomain
--
```

```

--                                         inserted
--
--*****CREATE TRIGGER TrigInsDmDdNumDomain
ON DmDdNumericDomain
FOR INSERT
AS
BEGIN
    DECLARE @attributeId      attIdType,
            @siteId          int,
            @hasIt           int

    BEGIN
        DECLARE insNumericDomainCS CURSOR FOR
            SELECT siteId, attributeId
            FROM inserted

        OPEN insNumericDomainCS

        FETCH insNumericDomainCS INTO
            @siteId,
            @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
            EXEC ProcCheckAttriExist @siteId = @siteId,
                                      @attributeId = @attributeId,
                                      @hasIt = @hasIt output
            IF @hasIt = 0
            BEGIN
                DELETE DmDdNumericDomain
                FROM DmDdNumericDomain s, inserted i
                WHERE s.siteId = i.siteId
                AND s.attributeId = i.attributeId

                RAISERROR 30515
                "No reference data exists for new record in DmDdNumericDomain,
inserting has been aborted"
                RETURN
            END

            FETCH insNumericDomainCS INTO
                @siteId,
                @attributeId
        END /* WHILE */
    END
END
RETURN
go

```

## **Trigger: TrigUpdDmDdNumericDomain**

### **Trigger Code**

```
--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigUpdDmDdNumericDomain
--
--TABLE ACCESSED:    DmDdCollECSAttributeXref
--                    DmDdNumericDomain
--                    inserted
--                    deleted
--
--*****
CREATE TRIGGER TrigUpdDmDdNumericDomain
ON DmDdNumericDomain
FOR UPDATE
AS
BEGIN
    DECLARE @attributeIdIns      attIdType,
            @siteIdIns        int,
            @attributeIdDel   attIdType,
            @siteIdDel        int

    BEGIN
        DECLARE updNumericDomainCS CURSOR FOR
            SELECT i.siteId, i.attributeId, d.siteId, d.attributeId
            FROM inserted i, deleted d
            WHERE i.siteId *= d.siteId
            AND i.attributeId *= d.attributeId

        OPEN updNumericDomainCS

        FETCH updNumericDomainCS INTO
            @siteIdIns,
            @attributeIdIns,
            @siteIdDel,
            @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
            IF @siteIdDel = null AND @attributeIdDel = null
            BEGIN

                IF (select count(*) FROM DmDdCollECSAttributeXref
                    WHERE siteId = @siteIdIns
                    AND attributeId = @attributeIdIns) = 0
                AND (SELECT count(*) FROM DmDdAdditionalAttributes
                    WHERE siteId = @siteIdIns)
```

```

        AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdEquivalentAttributes
              WHERE siteId = @siteIdIns
              AND attributeId = @attributeIdIns) = 0
    BEGIN
        DELETE DmDdNumericDomain
        FROM DmDdNumericDomain s, inserted i
        WHERE s.siteId = i.siteId
        AND s.attributeId = i.attributeId

        INSERT DmDdNumericDomain
        SELECT * from deleted

        RAISERROR 30515
        "No reference data exists for updated data in DmDdNumericDomain,
        updating has been aborted"
        RETURN
    END

    END /* if @siteIdDel = null and @attributeIdDel = null */

    FETCH updNumericDomainCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel

    END /* WHILE */
END
END
RETURN
go

```

### **Trigger: ti\_dmddoperationmodeclass**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddoperationmodeclass" for table "DmDdOperationModeClass" */
create trigger ti_dmddoperationmodeclass on DmDdOperationModeClass for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

```

```

/* Parent "DmDdInstrument" must exist when inserting a child in
"DmDdOperationModeClass" */
if update(InstrumentShortName)
begin
    select @numnull = (select count(*)
        from inserted
        where InstrumentShortName is null)
    if @numnull != @numrows
        if (select count(*)
            from DmDdInstrument t1, inserted t2
            where t1.InstrumentShortName = t2.InstrumentShortName) != @numrows - @numnull
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in
"DmDdOperationModeClass".'
            goto error
        end
    end
return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddplatformcharacteristic

### Trigger Code

```

/* Insert trigger "ti_dmddplatformcharacteristic" for table "DmDdPlatformCharacteristic" */
create trigger ti_dmddplatformcharacteristic on DmDdPlatformCharacteristic for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdPlatform" must exist when inserting a child in
    "DmDdPlatformCharacteristic" */
    if update(PlatformShortName)

```

```

begin
    select @numnull = (select count(*)
        from inserted
        where PlatformShortName is null)
    if @numnull != @numrows
        if (select count(*)
            from DmDdPlatform t1, inserted t2
            where t1.PlatformShortName = t2.PlatformShortName) != @numrows - @numnull
            begin
                select @errno = 30002,
                    @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
"DmDdPlatformCharacteristic".'
                goto error
            end
        end
    end

    return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## **Trigger: ti\_dmddplatformxref**

### **Trigger Code**

```

/* Insert trigger "ti_dmddplatformxref" for table "DmDdPlatformXref" */
create trigger ti_dmddplatformxref on DmDdPlatformXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdPlatform" must exist when inserting a child in "DmDdPlatformXref" */
    if update(PlatformShortName)
        begin
            if (select count(*)
                from DmDdPlatform t1, inserted t2
                where t1.PlatformShortName = t2.PlatformShortName) != @numrows
                begin

```

```

        select @errno = 30002,
               @errmsg = 'Parent does not exist in "DmDdPlatform". Cannot create child in
"DmDdPlatformXref".'
        goto error
      end
    end

/* Parent "DmDdECSCollection" must exist when inserting a child in "DmDdPlatformXref"
*/
if update(collectionId) or
   update(siteId)
begin
  if (select count(*)
      from DmDdECSCollection t1, inserted t2
      where t1.collectionId = t2.collectionId
      and t1.siteId = t2.siteId) != @numrows
  begin
    select @errno = 30002,
           @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdPlatformXref".'
    goto error
  end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

### **Trigger: ti\_dmddrangedatetime**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddrangedatetime" for table "DmDdRangeDateTime" */
create trigger ti_dmddrangedatetime on DmDdRangeDateTime for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

```

```

/* Parent "DmDdTemporal" must exist when inserting a child in "DmDdRangeDateTime" */
if update(collectionId) or
    update(siteId)
begin
    if (select count(*)
        from DmDdTemporal t1, inserted t2
        where t1.collectionId = t2.collectionId
        and t1.siteId = t2.siteId) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdTemporal". Cannot create child in
"DmDdRangeDateTime".'
        goto error
    end
end
return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddsensorcharacteristic

### Trigger Code

```

/* Insert trigger "ti_dmddsensorcharacteristic" for table "DmDdSensorCharacteristic" */
create trigger ti_dmddsensorcharacteristic on DmDdSensorCharacteristic for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorCharacteristic"
*/
    if update(SensorShortName)
    begin
        select @numnull = (select count(*)
            from inserted

```

```

        where SensorShortName is null)
if @numnull != @numrows
    if (select count(*)
        from DmDdSensor t1, inserted t2
        where t1.SensorShortName = t2.SensorShortName) != @numrows - @numnull
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorCharacteristic".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddsensorinstrumentxref

### Trigger Code

```

/* Insert trigger "ti_dmddsensorinstrumentxref" for table "DmDdSensorInstrumentXref" */
create trigger ti_dmddsensorinstrumentxref on DmDdSensorInstrumentXref for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdInstrument" must exist when inserting a child in
    "DmDdSensorInstrumentXref" */
    if update(InstrumentShortName)
    begin
        if (select count(*)
            from DmDdInstrument t1, inserted t2
            where t1.InstrumentShortName = t2.InstrumentShortName) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdInstrument". Cannot create child in

```

```

"DmDdSensorInstrumentXref".'
      goto error
    end
  end

/* Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorInstrumentXref"
*/
if update(SensorShortName)
begin
  if (select count(*)
    from DmDdSensor t1, inserted t2
    where t1.SensorShortName = t2.SensorShortName) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorInstrumentXref".'
    goto error
  end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## Trigger: ti\_dmddsensorxref

### Trigger Code

```

/* Insert trigger "ti_dmddsensorxref" for table "DmDdSensorXref" */
create trigger ti_dmddsensorxref on DmDdSensorXref for insert as
begin
  declare
    @numrows int,
    @numnull int,
    @errno int,
    @errmsg varchar(255)

  select @numrows = @@rowcount
  if @numrows = 0
    return

  /* Parent "DmDdSensor" must exist when inserting a child in "DmDdSensorXref" */
  if update(SensorShortName)
  begin

```

```

if (select count(*)
    from DmDdSensor t1, inserted t2
    where t1.SensorShortName = t2.SensorShortName) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdSensor". Cannot create child in
"DmDdSensorXref".'
        goto error
    end
end

/* Parent "DmDdECSCollection" must exist when inserting a child in "DmDdSensorXref" */
if update(collectionId) or
    update(siteId)
begin
    if (select count(*)
        from DmDdECSCollection t1, inserted t2
        where t1.collectionId = t2.collectionId
        and t1.siteId = t2.siteId) != @numrows
    begin
        select @errno = 30002,
            @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSensorXref".'
            goto error
        end
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: ti\_dmddsingletypecollection

### Trigger Code

```

/* Insert trigger "ti_dmddsingletypecollection" for table "DmDdSingleTypeCollection" */
create trigger ti_dmddsingletypecollection on DmDdSingleTypeCollection for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount

```

```

if @numrows = 0
    return

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdSingleTypeCollection" */
if update(collectionId) or
    update(siteId)
begin
if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
    and t1.siteId = t2.siteId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSingleTypeCollection".'
    goto error
end
end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

## Trigger: TrigInsDmDdSpatialDomain

### Trigger Code

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigInsDmDdSpatialDomain
--
--TABLE ACCESSED:   DmDdCollECSAttributeXref
--                  DmDdAdditionalAttributes
--                  DmDdSpatialDomain
--                  inserted
--
--*****

```

```

CREATE TRIGGER TrigInsDmDdSpatialDomain
ON DmDdSpatialDomain
FOR INSERT
AS

```

```

BEGIN
    DECLARE @attributeId      attIdType,
            @siteId          int,
            @hasIt           int

BEGIN
    DECLARE insSpatialDomainCS CURSOR FOR
        SELECT siteId, attributeId
        FROM inserted

    OPEN insSpatialDomainCS

    FETCH insSpatialDomainCS INTO
        @siteId,
        @attributeId

    WHILE @@sqlstatus != 2
    BEGIN
        EXEC ProcCheckAttriExist @siteId = @siteId,
                                  @attributeId = @attributeId,
                                  @hasIt = @hasIt output

        IF @hasIt = 0
        BEGIN
            DELETE DmDdSpatialDomain
            FROM DmDdSpatialDomain s, inserted i
            WHERE s.siteId = i.siteId
            AND s.attributeId = i.attributeId

            RAISERROR 30515
            "No reference data exists for new record in DmDdSpatialDomain, inserting
has been aborted"
            RETURN
        END

        FETCH insSpatialDomainCS INTO
            @siteId,
            @attributeId
    END /* WHILE */

END
END
RETURN
go

```

## **Trigger: TrigUpdDmDdSpatialDomain**

### **Trigger Code**

```
--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigUpdDmDdSpatialDomain
--
--TABLE ACCESSED:    DmDdCollECSAttributeXref
--                    DmDdSpatialDomain
--                    inserted
--                    deleted
--
--*****
CREATE TRIGGER TrigUpdDmDdSpatialDomain
ON DmDdSpatialDomain
FOR UPDATE
AS
BEGIN
    DECLARE @attributeIdIns      attIdType,
            @siteIdIns        int,
            @attributeIdDel   attIdType,
            @siteIdDel        int

    BEGIN
        DECLARE updSpatialDomainCS CURSOR FOR
            SELECT i.siteId, i.attributeId, d.siteId, d.attributeId
            FROM inserted i, deleted d
            WHERE i.siteId *= d.siteId
            AND i.attributeId *= d.attributeId

        OPEN updSpatialDomainCS

        FETCH updSpatialDomainCS INTO
            @siteIdIns,
            @attributeIdIns,
            @siteIdDel,
            @attributeIdDel

        WHILE @@sqlstatus != 2
        BEGIN
            IF @siteIdDel = null AND @attributeIdDel = null
            BEGIN

                IF (select count(*) FROM DmDdCollECSAttributeXref
                    WHERE siteId = @siteIdIns
                    AND attributeId = @attributeIdIns) = 0
                AND (SELECT count(*) FROM DmDdAdditionalAttributes
                    WHERE siteId = @siteIdIns)
```

```

        AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdEquivalentAttributes
              WHERE siteId = @siteIdIns
              AND attributeId = @attributeIdIns) = 0
    BEGIN
        DELETE DmDdSpatialDomain
        FROM DmDdSpatialDomain s, inserted i
        WHERE s.siteId = i.siteId
        AND s.attributeId = i.attributeId

        INSERT DmDdSpatialDomain
        SELECT * from deleted

        RAISERROR 30515
        "No reference data exists for updated data in DmDdSpatialDomain,
        updating has been aborted"
        RETURN
    END
    END /* if @siteIdDel = null and @attributeIdDel = null */

    FETCH updSpatialDomainCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel

    END /* WHILE */
END
END
RETURN
go

```

### **Trigger: ti\_dmddspatialkeywordclass**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddspatialkeywordclass" for table "DmDdSpatialKeywordClass" */
create trigger ti_dmddspatialkeywordclass on DmDdSpatialKeywordClass for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

```

```

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdSpatialKeywordClass" */
if update(collectionId) or
  update(siteId)
begin
  if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
      and t1.siteId = t2.siteId) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdSpatialKeywordClass".'
    goto error
  end
end
return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

## **Trigger: ti\_dmddstringdomain**

### **Trigger Code**

```

/* Insert trigger "ti_dmddstringdomain" for table "DmDdStringDomain" */
create trigger ti_dmddstringdomain on DmDdStringDomain for insert as
begin
declare
  @numrows int,
  @numnull int,
  @errno int,
  @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
  return

/* Parent "DmDdStringType" must exist when inserting a child in "DmDdStringDomain" */
if update(siteId) or
  update(attributeId)
begin
  if (select count(*)

```

```

from DmDdStringType t1, inserted t2
where t1.siteId = t2.siteId
and t1.attributeId = t2.attributeId) != @numrows
begin
    select @errno = 30002,
        @errmsg = 'Parent does not exist in "DmDdStringType". Cannot create child in
"DmDdStringDomain".'
        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddstringinstrumentchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddstringinstrumentchar" for table "DmDdStringInstrumentChar" */
create trigger ti_dmddstringinstrumentchar on DmDdStringInstrumentChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdInstrumentCharacteristic" must exist when inserting a child in
    "DmDdStringInstrumentChar" */
    if update(InstrumentCharacteristicName)
    begin
        if (select count(*)
            from DmDdInstrumentCharacteristic t1, inserted t2
            where t1.InstrumentCharacteristicName = t2.InstrumentCharacteristicName) !=
@numrows
            begin
                select @errno = 30002,
                    @errmsg = 'Parent does not exist in "DmDdInstrumentCharacteristic". Cannot create
child in "DmDdStringInstrumentChar".'
            end
    end
end

```

```

        goto error
    end
end

return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

```

### **Trigger: ti\_dmddstringplatformchar**

#### **Trigger Code**

```

/* Insert trigger "ti_dmddstringplatformchar" for table "DmDdStringPlatformChar" */
create trigger ti_dmddstringplatformchar on DmDdStringPlatformChar for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdPlatformCharacteristic" must exist when inserting a child in
    "DmDdStringPlatformChar" */
    if update(PlatformCharacteristicName)
    begin
        if (select count(*)
            from DmDdPlatformCharacteristic t1, inserted t2
            where t1.PlatformCharacteristicName = t2.PlatformCharacteristicName) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdPlatformCharacteristic". Cannot create
child in "DmDdStringPlatformChar".'
            goto error
        end
    end
    return

/* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction

```

```
end  
go
```

## Trigger: ti\_dmddstringsensorchar

### Trigger Code

```
/* Insert trigger "ti_dmddstringsensorchar" for table "DmDdStringSensorChar" */  
create trigger ti_dmddstringsensorchar on DmDdStringSensorChar for insert as  
begin  
    declare  
        @numrows int,  
        @numnull int,  
        @errno int,  
        @errmsg varchar(255)  
  
    select @numrows = @@rowcount  
    if @numrows = 0  
        return  
  
    /* Parent "DmDdSensorCharacteristic" must exist when inserting a child in  
    "DmDdStringSensorChar" */  
    if update(SensorCharacteristicName)  
    begin  
        if (select count(*)  
            from DmDdSensorCharacteristic t1, inserted t2  
            where t1.SensorCharacteristicName = t2.SensorCharacteristicName) != @numrows  
        begin  
            select @errno = 30002,  
                  @errmsg = 'Parent does not exist in "DmDdSensorCharacteristic". Cannot create  
child in "DmDdStringSensorChar".'  
            goto error  
        end  
    end  
  
    return  
  
/* Errors handling */  
error:  
    raiserror @errno @errmsg  
    rollback transaction  
end  
go
```

## **Trigger: TrigInsDmDdStringType**

### **Trigger Code**

```
--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigInsDmDdStringType
--
--TABLE ACCESSED:    DmDdCollECSAttributeXref
--                   DmDdAdditionalAttributes
--                   DmDdStringType
--                   inserted
--
--*****
CREATE TRIGGER TrigInsDmDdStringType
ON DmDdStringType
FOR INSERT
AS
BEGIN
    DECLARE @attributeId      attIdType,
            @siteId          int,
            @hasIt           int

    BEGIN
        DECLARE insStringTypeCS CURSOR FOR
            SELECT siteId, attributeId
            FROM inserted

        OPEN insStringTypeCS

        FETCH insStringTypeCS INTO
            @siteId,
            @attributeId

        WHILE @@sqlstatus != 2
        BEGIN
            EXEC ProcCheckAttriExist @siteId = @siteId,
                                      @attributeId = @attributeId,
                                      @hasIt = @hasIt output
            IF @hasIt = 0
            BEGIN
                DELETE DmDdStringType
                FROM DmDdStringType s, inserted i
                WHERE s.siteId = i.siteId
                AND s.attributeId = i.attributeId

                RAISERROR 30515
                "No reference data exists for new record in DmDdStringType, inserting has
been aborted"
            END
        END
    END
END
```

```

        RETURN
    END

    FETCH insStringTypeCS INTO
        @siteId,
        @attributeId
    END /* WHILE */
END
END
RETURN
go

```

### **Trigger: TrigUpdDmDdStringType**

#### **Trigger Code**

```

--*****
--BEGIN PROLOG
--
--TRIGGER NAME:      TrigUpdDmDdStringType
--
--TABLE ACCESSED:    DmDdCollECSAttributeXref
--                    DmDdStringType
--                    inserted
--                    deleted
--
--*****
CREATE TRIGGER TrigUpdDmDdStringType
ON DmDdStringType
FOR UPDATE
AS
BEGIN
    DECLARE @attributeIdIns          attIdType,
            @siteIdIns           int,
            @attributeIdDel       attIdType,
            @siteIdDel           int

    BEGIN
        DECLARE updStringTypeCS CURSOR FOR
            SELECT i.siteId, i.attributeId, d.siteId, d.attributeId
            FROM inserted i, deleted d
            WHERE i.siteId *= d.siteId
            AND i.attributeId *= d.attributeId

        OPEN updStringTypeCS

        FETCH updStringTypeCS INTO
            @siteIdIns,
            @attributeIdIns,

```

```

@siteIdDel,
@attributeIdDel

WHILE @@sqlstatus != 2
BEGIN
    IF @siteIdDel = null AND @attributeIdDel = null
    BEGIN

        IF (select count(*) FROM DmDdCollECSAttributeXref
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdAdditionalAttributes
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        AND (SELECT count(*) FROM DmDdEquivalentAttributes
            WHERE siteId = @siteIdIns
            AND attributeId = @attributeIdIns) = 0
        BEGIN
            DELETE DmDdStringType
            FROM DmDdStringType s, inserted i
            WHERE s.siteId = i.siteId
            AND s.attributeId = i.attributeId

            INSERT DmDdStringType
            SELECT * from deleted

            RAISERROR 30515
            "No reference data exists for updated data in DmDdStringType, updating
has been aborted"
            RETURN
        END
    END /* if @siteIdDel = null and @attributeIdDel = null */

    FETCH updStringTypeCS INTO
        @siteIdIns,
        @attributeIdIns,
        @siteIdDel,
        @attributeIdDel

    END /* WHILE */
END
END
RETURN
go

```

## **Trigger: ti\_dmddtemporal**

### **Trigger Code**

```
/* Insert trigger "ti_dmddtemporal" for table "DmDdTemporal" */
create trigger ti_dmddtemporal on DmDdTemporal for insert as
begin
    declare
        @numrows int,
        @numnull int,
        @errno int,
        @errmsg varchar(255)

    select @numrows = @@rowcount
    if @numrows = 0
        return

    /* Parent "DmDdECSCollection" must exist when inserting a child in "DmDdTemporal" */
    if update(collectionId) or
        update(siteId)
    begin
        if (select count(*)
            from DmDdECSCollection t1, inserted t2
            where t1.collectionId = t2.collectionId
            and t1.siteId = t2.siteId) != @numrows
        begin
            select @errno = 30002,
                @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdTemporal".'
            goto error
        end
    end
    return

    /* Errors handling */
error:
    raiserror @errno @errmsg
    rollback transaction
end
go
```

## **Trigger: ti\_dmddtemporalkeywordclass**

### **Trigger Code**

```
/* Insert trigger "ti_dmddtemporalkeywordclass" for table "DmDdTemporalKeywordClass" */
create trigger ti_dmddtemporalkeywordclass on DmDdTemporalKeywordClass for insert as
begin
```

```

declare
  @numrows int,
  @numnull int,
  @errno int,
  @errmsg varchar(255)

select @numrows = @@rowcount
if @numrows = 0
  return

/* Parent "DmDdECSCollection" must exist when inserting a child in
"DmDdTemporalKeywordClass" */
if update(collectionId) or
  update(siteId)
begin
  if (select count(*)
    from DmDdECSCollection t1, inserted t2
    where t1.collectionId = t2.collectionId
    and t1.siteId = t2.siteId) != @numrows
  begin
    select @errno = 30002,
      @errmsg = 'Parent does not exist in "DmDdECSCollection". Cannot create child in
"DmDdTemporalKeywordClass".'
    goto error
  end
end
end

return

/* Errors handling */
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

```

#### 4.1.10 Stored Procedures

Sybase also includes support for business policy via the use of stored procedures. Stored procedures are typically used to capture a set of activities or checks that will be performed on the database repeatedly to enforce business policy and maintain data integrity. Stored procedures are parsed and complied SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. A listing of each the stored procedures in the DM database is given here. A brief definition of each of these stored procedures follows.

## Procedure List

Name	Description
ProcCheckAttriExist	TBS
ProcDeleteAdditionalAttributes	TBS
ProcDeleteAttributeRelated	TBS
ProcDeleteCollECSAttributeXref	TBS
ProcDeleteCollection	TBS
ProcDeleteECSAttributes	TBS
ProcDeleteEquivalentAttributes	TBS
ProcDeleteInfoMgr	TBS
ProcDeleteInstrument	TBS
ProcDeletePlatform	TBS
ProcDeleteSensor	TBS

### Procedure: ProcCheckAttriExist

#### Code

```

CREATE PROC ProcCheckAttriExist
    (@siteId      int,
     @attributeId attIdType,
     @hasIt        int          output)
AS
BEGIN
    IF (SELECT count(*) FROM DmDdCollECSAttributeXref
        WHERE siteId = @siteId
        AND attributeId = @attributeId) = 0
    BEGIN
        IF (SELECT count(*) FROM DmDdAdditionalAttributes
            WHERE siteId = @siteId
            AND attributeId = @attributeId) = 0
        BEGIN
            IF (SELECT count(*) FROM DmDdEquivalentAttributes
                WHERE siteId = @siteId
                AND attributeId = @attributeId) = 0
            BEGIN
                SELECT @hasIt = 0
                RETURN
            END
        END
    END
    END
    SELECT @hasIt = 1
    RETURN
END
go

```

## **Procedure: ProcDeleteAdditionalAttributes**

### **Code**

```
--||||||||||||||||||||||||||||||||||||||||  
-- Name : ProcDeleteAdditionalAttributes  
--  
-- Description:  
-- Delete all related records in all referenceing tables  
-- when delete a DmDdAdditionalAttributes record  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||||||||||||  
  
create proc ProcDeleteAdditionalAttributes(  
    @attributeId int,  
    @siteId      int,  
    @error       int      output)  
as  
    BEGIN TRANSACTION  
  
        IF (SELECT count(*) FROM DmDdAdditionalAttributes  
            WHERE attributeId = @attributeId  
                AND siteId = @siteId) = 0  
            BEGIN  
                raiserror 30515  
                "No such AdditionalAttributes exists"  
                ROLLBACK TRANSACTION  
                RETURN  
            END  
        else  
            BEGIN --START DELETE  
  
                EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,  
                    @siteId = @siteId,  
                    @error = @error output  
                IF @error != 0  
                    BEGIN  
                        raiserror 30515  
                        "An error occurred when delete Attribute Related"  
                        ROLLBACK TRANSACTION  
                        RETURN  
                    END  
            END  
        --  
        --DmDdAdditionalAttributes  
        --  
            DELETE DmDdAdditionalAttributes  
            WHERE attributeId = @attributeId  
                AND siteId = @siteId
```

```

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdAdditionalAttributes"
    ROLLBACK TRANSACTION
    RETURN
END
END -- end of START DELETE

COMMIT TRANSACTION

RETURN
go

```

### **Procedure: ProcDeleteAttributeRelated**

#### **Code**

```

--/////////////////////////////////////////////////////////////////////////
-- Name : ProcDeleteAttributeRelated
--
-- Description:
--
-- Delete all related records in all referencing tables
-- for delete a DmDdColleCSAttributeXref record
-- or delete a DmDdAdditionalAttributes record
-- or delete a DmDdEquivalentAttributes record
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////

create proc ProcDeleteAttributeRelated(
    @attributeId int,
    @siteId      int,
    @error       int      output)
as
--
--DmDdAttributeXref
--
    DELETE DmDdAttributeXref
    WHERE attributeId = @attributeId
        AND siteId = @siteId
    OR attributeIdRef = @attributeId
        AND siteIdRef = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdAttributeXref"
    END

```

```

        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdSpatialDomain
--
    DELETE DmDdSpatialDomain
    WHERE attributeId = @attributeId
        AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdSpatialDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdNumericDomain
--
    DELETE DmDdNumericDomain
    WHERE attributeId = @attributeId
        AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdNumericDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdDateTimeDomain
--
    DELETE DmDdDateTimeDomain
    WHERE attributeId = @attributeId
        AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdDateTimeDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdKeywordAliasXref
--
    DELETE DmDdKeywordAliasXref

```

```

WHERE attributeId = @attributeId
    AND siteId = @siteId
OR attributeIdRef = @attributeId
    AND siteIdRef = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdKeywordAliasXref"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdStringDomain
DELETE DmDdStringDomain
WHERE attributeId = @attributeId
    AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdStringDomain"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdStringType
DELETE DmDdStringType
WHERE attributeId = @attributeId
    AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdStringType"
    ROLLBACK TRANSACTION
    RETURN
END

RETURN
go

```

## **Procedure: ProcDeleteCollECSAttributeXref**

### **Code**

```
--|||||||||||||||||||||||||||||||||||||||  
-- Name : ProcDeleteCollECSAttributeXref  
--  
-- Description:  
-- Delete all related records in all referenceing tables  
-- when delete a DmDdCollECSAttributeXref record  
--  
-- Implementation Notes:  
--|||||||||||||||||||||||||||||||||||  
  
create proc ProcDeleteCollECSAttributeXref(  
    @attributeId int,  
    @siteId      int,  
    @error       int      output)  
as  
    BEGIN TRANSACTION  
  
    IF (SELECT count(*) FROM DmDdCollECSAttributeXref  
        WHERE attributeId = @attributeId  
        AND siteId = @siteId) = 0  
        BEGIN  
            raiserror 30515  
            "No such CollECSAttributeXref exists"  
            ROLLBACK TRANSACTION  
            RETURN  
        END  
    else  
        BEGIN --START DELETE  
  
            EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,  
                @siteId = @siteId,  
                @error = @error output  
            IF @error != 0  
                BEGIN  
                    raiserror 30515  
                    "An error occurred when delete Attribute Related"  
                    ROLLBACK TRANSACTION  
                    RETURN  
                END  
        END  
    --  
    --DmDdCollECSAttributeXref  
    --  
        DELETE DmDdCollECSAttributeXref  
        WHERE attributeId = @attributeId
```

```

        AND siteId = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
            raiserror 30515
            "An error occurred when delete DmDdCollECSAttributeXref"
            ROLLBACK TRANSACTION
            RETURN
        END
    END -- end of START DELETE

    COMMIT TRANSACTION

    RETURN
go

```

### **Procedure: ProcDeleteCollection**

#### **Code**

```

--///////////
-- Name : ProcDeleteCollection
--
-- Description:
--
-- Delete all related records in all referencing tables
-- when delete a DmDdECSCollection record
--
-- Implementation Notes:
--///////////

create proc ProcDeleteCollection(
    @collectionId int,
    @siteId      int,
    @error       int      output)
as
    BEGIN TRANSACTION

    IF (SELECT count(*) FROM DmDdECSCollection
        WHERE collectionId = @collectionId
        AND siteId = @siteId) = 0
        BEGIN
            raiserror 30515
            "No such collection exists"
            ROLLBACK TRANSACTION
            RETURN
        END
    else
        BEGIN --START DELETE

```

```

--DmDdTemporalKeywordClass
--
DELETE DmDdTemporalKeywordClass
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdTemporalKeywordClass"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdSpatialKeywordClass
--
DELETE DmDdSpatialKeywordClass
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdSpatialKeywordClass"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdCollectionXref
--
DELETE DmDdCollectionXref
WHERE collectionIdRef = @collectionId
AND siteIdRef = @siteId
OR collectionIdAgg = @collectionId
AND siteIdAgg = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdCollectionXref"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdSingleTypeCollection
--
DELETE DmDdSingleTypeCollection
WHERE collectionId = @collectionId

```

```

AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdSingleTypeCollection"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdMultipleCollection
-- DELETE DmDdMultipleCollection
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdMultipleCollection"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdKeywordParameterXref
-- DELETE DmDdKeywordParameterXref
WHERE keywordId =
    (SELECT keywordId FROM DmDdECSCollKeywordXref
     WHERE collectionId = @collectionId
     AND siteId = @siteId
    )
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdKeywordParameterXref"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdECSCollKeywordXref
-- DELETE DmDdECSCollKeywordXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error

```

```

IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdECSCollKeywordXref"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdSensorXref
--
DELETE DmDdSensorXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdSensorXref"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdInstrumentXref
--
DELETE DmDdInstrumentXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdInstrumentXref"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdPlatformXref
--
DELETE DmDdPlatformXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdPlatformXref"
    ROLLBACK TRANSACTION
    RETURN
END

```

```

--DmDdSpatialDomain
--
    DELETE DmDdSpatialDomain
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdSpatialDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdNumericDomain
--
    DELETE DmDdNumericDomain
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdNumericDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdDateTimeDomain
--
    DELETE DmDdDateTimeDomain
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdDateTimeDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdKeywordAliasXref
--
    DELETE DmDdKeywordAliasXref
    WHERE collectionId = @collectionId
    AND siteId = @siteId
    OR collectionIdRef = @collectionId

```

```

        AND siteIdRef = @siteId

        SELECT @error = @@error
        IF @error != 0
        BEGIN
            raiserror 30515
            "An error occurred when delete DmDdKeywordAliasXref"
            ROLLBACK TRANSACTION
            RETURN
        END
    --
--DmDdStringDomain
--
    DELETE DmDdStringDomain
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdStringDomain"
        ROLLBACK TRANSACTION
        RETURN
    END
    --
--DmDdStringType
--
    DELETE DmDdStringType
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdStringType"
        ROLLBACK TRANSACTION
        RETURN
    END
    --
--DmDdAttributeXref
--
    DELETE DmDdAttributeXref
    WHERE collectionId = @collectionId
    AND siteId = @siteId
    OR collectionIdRef = @collectionId
    AND siteIdRef = @siteId

    SELECT @error = @@error
    IF @error != 0
    BEGIN

```

```

raiserror 30515
"An error occurred when delete DmDdAttributeXref"
ROLLBACK TRANSACTION
RETURN
END

--DmDdCollECSAttributeXref
--
DELETE DmDdCollECSAttributeXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
raiserror 30515
"An error occurred when delete DmDdCollECSAttributeXref"
ROLLBACK TRANSACTION
RETURN
END

--DmDdAdditionalAttributes
--
DELETE DmDdAdditionalAttributes
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
raiserror 30515
"An error occurred when delete DmDdAdditionalAttributes"
ROLLBACK TRANSACTION
RETURN
END

--DmDdEquivalentAttributes
--
DELETE DmDdEquivalentAttributes
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
raiserror 30515
"An error occurred when delete DmDdEquivalentAttributes"
ROLLBACK TRANSACTION
RETURN
END

--DmDdInfoMgrCollXref

```

```

-- DELETE DmDdInfoMgrCollXref
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdInfoMgrCollXref"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdBoundingRectangle
-- DELETE DmDdBoundingRectangle
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdBoundingRectangle"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdRangeDateTime
-- DELETE DmDdRangeDateTime
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdRangeDateTime"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdTTemporal
-- DELETE DmDdTTemporal
WHERE collectionId = @collectionId
AND siteId = @siteId

SELECT @error = @@error
IF @error != 0

```

```

        BEGIN
            raiserror 30515
            "An error occurred when delete DmDdTemporal"
            ROLLBACK TRANSACTION
            RETURN
        END
    --
--DmDdECSCollection
--
    DELETE DmDdECSCollection
    WHERE collectionId = @collectionId
    AND siteId = @siteId

    SELECT @error = @@error
    IF @error != 0
        BEGIN
            raiserror 30515
            "An error occurred when delete DmDdECSCollection"
            ROLLBACK TRANSACTION
            RETURN
        END
    END -- end of START DELETE

    COMMIT TRANSACTION

RETURN
go

```

### **Procedure: ProcDeleteECSAttributes**

#### **Code**

```

--///////////
-- Name : ProcDeleteECSAttributes
--
-- Description:
--
-- Delete all related records in all referencing tables
-- when delete a DmDdECSAttributes record
--
-- Implementation Notes:
--///////////

create proc ProcDeleteECSAttributes(
    @attributeName      char(32),
    @error      int      output)
as
    BEGIN TRANSACTION

    IF (SELECT count(*) FROM DmDdECSAttributes
        WHERE ECSAttributeName = @attributeName ) = 0
    BEGIN

```

```

raiserror 30515
"No such ECSAttribute exists"
    ROLLBACK TRANSACTION
    RETURN
END

else

BEGIN --START DELETE

declare axref_cursor cursor for
    SELECT attributeId, siteId
    FROM DmDdCollECSAttributeXref
    WHERE ECSAttributeName = @attributeName

open axref_cursor

declare @attributeId int,
        @siteId      int

fetch axref_cursor into
    @attributeId,
    @siteId

while @@sqlstatus != 2
begin

    EXEC ProcDeleteCollECSAttributeXref
        @attributeId = @attributeId,
        @siteId = @siteId,
        @error = @error output

    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete CollECSAttributeXref"
        ROLLBACK TRANSACTION
        RETURN
    END

    fetch axref_cursor into
        @attributeId,
        @siteId

end

DELETE DmDdECSAttributes
WHERE ECSAttributeName = @attributeName

SELECT @error = @@error
IF @error != 0
BEGIN

```

```

raiserror 30515
"An error occurred when delete DmDdECSAttributes"
ROLLBACK TRANSACTION
RETURN
END

END -- end of START DELETE

COMMIT TRANSACTION

RETURN
go

```

### **Procedure: ProcDeleteEquivalentAttributes**

#### **Code**

```

--|||||||||||||||||||||||||||||||||||||||||
-- Name : ProcDeleteEquivalentAttributes
--
-- Description:
--
-- Delete all related records in all referencing tables
-- when delete a DmDdEquivalentAttributes record
--
-- Implementation Notes:
--||||||||||||||||||||||||||||||||||

create proc ProcDeleteEquivalentAttributes(
    @attributeId int,
    @siteId      int,
    @error       int      output)
as
BEGIN TRANSACTION

IF (SELECT count(*) FROM DmDdEquivalentAttributes
    WHERE attributeId = @attributeId
        AND siteId = @siteId) = 0
    BEGIN
        raiserror 30515
        "No such EquivalentAttributes exists"
        ROLLBACK TRANSACTION
        RETURN
    END
else
BEGIN --START DELETE

    EXEC ProcDeleteAttributeRelated @attributeId = @attributeId,
                                    @siteId = @siteId,
                                    @error = @error output

```

```

IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete Attribute Related"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdEquivalentAttributes
--
DELETE DmDdEquivalentAttributes
WHERE attributeId = @attributeId
    AND siteId = @siteId

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdEquivalentAttributes"
    ROLLBACK TRANSACTION
    RETURN
END
END -- end of START DELETE

COMMIT TRANSACTION

RETURN
go

```

### **Procedure: ProcDeleteInfoMgr**

#### **Code**

```

--/////////////////////////////////////////////////////////////////////////
-- Name : ProcDeleteInfoMgr
--
-- Description:
--
-- Delete all related records in all referencing tables
-- when delete a DmDdInfoMgr record
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////

create proc ProcDeleteInfoMgr(
    @infoMgrName      char(20),
    @error           int      output)
as
BEGIN TRANSACTION

IF (SELECT count(*) FROM DmDdInfoMgr
    WHERE infoMgrName = @infoMgrName

```

```

        ) = 0
    BEGIN
raiserror 30515
"No such InfoMgr exists"
        ROLLBACK TRANSACTION
        RETURN
    END
else
    BEGIN --START DELETE
--
--DmDdInfoMgrCollXref
--
    DELETE DmDdInfoMgrCollXref
    WHERE infoMgrName = @infoMgrName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdInfoMgrCollXref"
        ROLLBACK TRANSACTION
        RETURN
    END
--
--DmDdInfoMgrXref
--
    DELETE DmDdInfoMgrXref
    WHERE infoMgrName = @infoMgrName
    OR infoMgrParentName = @infoMgrName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdInfoMgrXref"
        ROLLBACK TRANSACTION
        RETURN
    END
--
--DmDdInfoMgr
--
    DELETE DmDdInfoMgr
    WHERE infoMgrName = @infoMgrName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdInfoMgr"
        ROLLBACK TRANSACTION

```

```

        RETURN
    END
END -- end of START DELETE

COMMIT TRANSACTION

```

RETURN  
go

### **Procedure: ProcDeleteInstrument**

#### **Code**

```

--///////////////////////////////
-- Name : ProcDeleteInstrument
--
-- Description:
--
-- Delete all related records in all referenceing tables
-- when delete a DmDdECSInstrument record
--
-- Implementation Notes:
--////////////////////

create proc ProcDeleteInstrument(
    @instrumentShortName      char(20),
    @error                      int      output)
as
    BEGIN TRANSACTION

    IF (SELECT count(*) FROM DmDdInstrument
        WHERE InstrumentShortName = @instrumentShortName
        ) = 0
        BEGIN
        raiserror 30515
        "No such Instrument exists"
        ROLLBACK TRANSACTION
        RETURN
    END

    else

        BEGIN --START DELETE
        --
        --DmDdInstrumentXref
        --
        DELETE DmDdInstrumentXref
        WHERE InstrumentShortName = @instrumentShortName

        SELECT @error = @@error
        IF @error != 0
        BEGIN

```

```

raiserror 30515
"An error occurred when delete DmDdInstrumentXref"
ROLLBACK TRANSACTION
RETURN
END

--DmDdInstrumentPlatformXref
--
DELETE DmDdInstrumentPlatformXref
WHERE InstrumentShortName = @instrumentShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdInstrumentPlatformXref"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdOperationModeClass
--
DELETE DmDdOperationModeClass
WHERE InstrumentShortName = @instrumentShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdOperationModeClass"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdSensorInstrumentXref
--
DELETE DmDdSensorInstrumentXref
WHERE InstrumentShortName = @instrumentShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdSensorInstrumentXref"
    ROLLBACK TRANSACTION
    RETURN
END

--DmDdIntInstrumentChar
--
DELETE DmDdIntInstrumentChar
WHERE InstrumentCharacteristicName in

```

```

        (SELECT InstrumentCharacteristicName
         FROM DmDdInstrumentCharacteristic
        WHERE InstrumentShortName = @instrumentShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdIntInstrumentChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdFloatInstrumentChar
--
DELETE DmDdFloatInstrumentChar
WHERE InstrumentCharacteristicName in
    (SELECT InstrumentCharacteristicName
     FROM DmDdInstrumentCharacteristic
     WHERE InstrumentShortName = @instrumentShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdFloatInstrumentChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdDateInstrumentChar
--
DELETE DmDdDateInstrumentChar
WHERE InstrumentCharacteristicName in
    (SELECT InstrumentCharacteristicName
     FROM DmDdInstrumentCharacteristic
     WHERE InstrumentShortName = @instrumentShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdDateInstrumentChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdStringInstrumentChar
--
DELETE DmDdStringInstrumentChar
WHERE InstrumentCharacteristicName in
    (SELECT InstrumentCharacteristicName

```

```

        FROM DmDdInstrumentCharacteristic
        WHERE InstrumentShortName = @instrumentShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdStringInstrumentChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdInstrumentCharacteristic
--
DELETE DmDdInstrumentCharacteristic
WHERE InstrumentShortName = @instrumentShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdInstrumentCharacteristic"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdInstrument
--
DELETE DmDdInstrument
WHERE InstrumentShortName = @instrumentShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdInstrument"
    ROLLBACK TRANSACTION
    RETURN
END
END -- end of START DELETE

COMMIT TRANSACTION

RETURN
go

```

### **Procedure: ProcDeletePlatform**

#### **Code**

```
--/////////////////////////////////////////////////////////////////////////
-- Name : ProcDeletePlatform
```

```

-- Description:
--
-- Delete all related records in all referencing tables
-- when delete a DmDdECSPlatform record
--
-- Implementation Notes:
--||||||||||||||||||||||||||||||||||||||

create proc ProcDeletePlatform(
    @platformShortName char(20),
    @error                int      output)
as
    BEGIN TRANSACTION

        IF (SELECT count(*) FROM DmDdPlatform
            WHERE PlatformShortName = @platformShortName
            ) = 0
        BEGIN
            raiserror 30515
            "No such Platform exists"
            ROLLBACK TRANSACTION
            RETURN
        END
    else
        BEGIN --START DELETE
        --
        --DmDdPlatformXref
        --
            DELETE DmDdPlatformXref
            WHERE PlatformShortName = @platformShortName

            SELECT @error = @@error
            IF @error != 0
            BEGIN
                raiserror 30515
                "An error occurred when delete DmDdPlatformXref"
                ROLLBACK TRANSACTION
                RETURN
            END
        --
        --DmDdInstrumentPlatformXref
        --
            DELETE DmDdInstrumentPlatformXref
            WHERE PlatformShortName = @platformShortName

            SELECT @error = @@error
            IF @error != 0
            BEGIN
                raiserror 30515
            END
        END
    END

```

```

        "An error occured when delete DmDdInstrumentPlatformXref"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdIntPlatformChar
--
    DELETE DmDdIntPlatformChar
    WHERE PlatformCharacteristicName in
        (SELECT PlatformCharacteristicName
         FROM DmDdPlatformCharacteristic
         WHERE PlatformShortName = @platformShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occured when delete DmDdIntPlatformChar"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdFloatPlatformChar
--
    DELETE DmDdFloatPlatformChar
    WHERE PlatformCharacteristicName in
        (SELECT PlatformCharacteristicName
         FROM DmDdPlatformCharacteristic
         WHERE PlatformShortName = @platformShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occured when delete DmDdFloatPlatformChar"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdDatePlatformChar
--
    DELETE DmDdDatePlatformChar
    WHERE PlatformCharacteristicName in
        (SELECT PlatformCharacteristicName
         FROM DmDdPlatformCharacteristic
         WHERE PlatformShortName = @platformShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occured when delete DmDdDatePlatformChar"
    END

```

```

        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdStringPlatformChar
--
    DELETE DmDdStringPlatformChar
    WHERE PlatformCharacteristicName in
        (SELECT PlatformCharacteristicName
         FROM DmDdPlatformCharacteristic
         WHERE PlatformShortName = @platformShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdStringPlatformChar"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdPlatformCharacteristic
--
    DELETE DmDdPlatformCharacteristic
    WHERE PlatformShortName = @platformShortName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdPlatformCharacteristic"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdPlatform
--
    DELETE DmDdPlatform
    WHERE PlatformShortName = @platformShortName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdPlatform"
        ROLLBACK TRANSACTION
        RETURN
    END
END -- end of START DELETE

COMMIT TRANSACTION

```

```
RETURN  
go
```

### Procedure: ProcDeleteSensor

#### Code

```
--||||||||||||||||||||||||||||||||  
-- Name : ProcDeleteSensor  
--  
-- Description:  
-- Delete all related records in all referencing tables  
-- when delete a DmDdECSSensor record  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||||||||  
  
create proc ProcDeleteSensor(  
    @sensorShortName  char(20),  
    @error           int      output)  
as  
    BEGIN TRANSACTION  
  
        IF (SELECT count(*) FROM DmDdSensor  
            WHERE SensorShortName = @sensorShortName  
            ) = 0  
            BEGIN  
                raiserror 30515  
                "No such Sensor exists"  
                ROLLBACK TRANSACTION  
                RETURN  
            END  
  
        else  
  
            BEGIN --START DELETE  
--  
--DmDdSensorXref  
--  
            DELETE DmDdSensorXref  
            WHERE SensorShortName = @sensorShortName  
  
            SELECT @error = @@error  
            IF @error != 0  
            BEGIN  
                raiserror 30515  
                "An error occurred when delete DmDdSensorXref"  
                ROLLBACK TRANSACTION  
                RETURN  
            END  
--
```

```

--DmDdSensorInstrumentXref
--
    DELETE DmDdSensorInstrumentXref
    WHERE SensorShortName = @sensorShortName

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdSensorInstrumentXref"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdIntSensorChar
--
    DELETE DmDdIntSensorChar
    WHERE SensorCharacteristicName in
        (SELECT SensorCharacteristicName
         FROM DmDdSensorCharacteristic
         WHERE SensorShortName = @sensorShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdIntSensorChar"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdFloatSensorChar
--
    DELETE DmDdFloatSensorChar
    WHERE SensorCharacteristicName in
        (SELECT SensorCharacteristicName
         FROM DmDdSensorCharacteristic
         WHERE SensorShortName = @sensorShortName)

    SELECT @error = @@error
    IF @error != 0
    BEGIN
        raiserror 30515
        "An error occurred when delete DmDdFloatSensorChar"
        ROLLBACK TRANSACTION
        RETURN
    END
--

--DmDdDateSensorChar
--
    DELETE DmDdDateSensorChar
    WHERE SensorCharacteristicName in

```

```

        (SELECT SensorCharacteristicName
         FROM DmDdSensorCharacteristic
        WHERE SensorShortName = @sensorShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdDateSensorChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdStringSensorChar
-- DELETE DmDdStringSensorChar
WHERE SensorCharacteristicName in
    (SELECT SensorCharacteristicName
     FROM DmDdSensorCharacteristic
     WHERE SensorShortName = @sensorShortName)

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdStringSensorChar"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdSensorCharacteristic
-- DELETE DmDdSensorCharacteristic
WHERE SensorShortName = @sensorShortName

SELECT @error = @@error
IF @error != 0
BEGIN
    raiserror 30515
    "An error occurred when delete DmDdSensorCharacteristic"
    ROLLBACK TRANSACTION
    RETURN
END
--DmDdSensor
-- DELETE DmDdSensor
WHERE SensorShortName = @sensorShortName

SELECT @error = @@error
IF @error != 0
BEGIN

```

```
raiserror 30515
      "An error occurred when delete DmDdSensor"
      ROLLBACK TRANSACTION
      RETURN
END
END -- end of START DELETE

COMMIT TRANSACTION

RETURN
go
```

## 4.2 File Usage

There are cases when the implementation of a persistent data requirement is better suited to a flat file than to a database table. A typical example of such data is system configuration information. System configuration information is fairly static and usually has no explicit relationship to other data in the enterprise. Another common use of files in ECS is as an interface mechanism between ECS and the external world. FIILSUB file usage is detailed in this section via file definitions, attribute definitions, and attribute domain definitions.

### 4.2.1 Files Definitions

A listing of each the files in the DM database is given here. A brief definition of each of these files follows.

### 4.2.2 Attributes

Brief definitions of each of the attributes present in the files defined above are contained herein.

### 4.2.3 Attribute Domains

Domains represent the ranges of valid values allowed for a given file attribute. Attributes domains for each of the attributes defined above are given here.

# **5 Performance and Tuning Factors**

---

## **5.1 Indexes**

An index provides a means of locating a row in a table based on the value of specific columns, without having to scan each row in the table. If used appropriately, indexes can significantly increase data retrieval. Sybase allows the definition of two types of indexes, clustered and non-clustered. In a clustered index, the rows in a table are physically stored in the sort order determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in order. Non-clustered indexes differ from their clustered counterpart, in that data is not physically stored in sort order. Only one clustered index may be defined per table. A list of all the indexes defined against tables in the DM database is given here along with a description of each index..

## **5.2 Segments**

Sybase supports the definition of segments. A segment is a named pointer to a storage device or devices. Segments are used to manually place database objects onto particular storage devices. All segments explicitly defined in the DM database are described herein.

## **5.3 Named Caches**

A cache is a block of memory that is used by Sybase to house data pages that are currently being accessed. A named cache is a named block of memory that the SQL server can use to house frequently accessed tables. Assigning a table to cache causes it to be loaded into memory. This greatly increases performance by eliminating the time expense normally associated with disk i/o. Named caches used in the DM databases are described herein.

This page intentionally left blank.

## 6. Database Security

---

### 6.1 Initial Users

Upon initial installation the following users will have access to DM database. The level of access is limited to that associated with their assigned group and/or role. A complete definition of each of these groups and roles is given below.

### 6.2 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. Several group have been defined in the DM database upon initial installation. A definition of each of these groups is contained herein.

### 6.3 Roles

Roles were introduced in Sybase 10 to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows as well as a description of the types of activities that may be performed by each role.

**System Administrator** (sa\_role) - This role is used to grant a specific user to permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

**Site Security Officer** (sso\_role) - This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except sa\_role

**Operator** (oper\_role) - This role is used to grant a specific user the permissions needed to manage backup and recovery of the database including;

- dumping transactions and databases
- loading transactions and databases

**Navigator** (navigator\_role) -This role is used to grant a specific user the permissions needed to manage the navigation server.

**Replication** (replication\_role) - - This role is used to grant a specific user the permissions needed to manage the replication server.

**Sybase Technical Support** (sybase\_ts\_role) - This role is used to grant a specific user the permissions needed to perform database consistency checker (dbcc), a sybase supplied utility, commands that are considered outside of the realm of normal system administrator activities.

## 6.4 Object Permissions

TBS

# **7. Replication**

---

## **7.1 Replication Overview**

Replication as the name implies is a set of Sybase products that allow replication of data from one database to another. The DM database employs replication to support its DATA DISTRIBUTION requirements. In order for replication to be accomplished the data source must define the tables and columns that may be replicated to a data recipient. These definitions are referred to replication definitions. In the same manner a data recipient must specify the replication definitions in which he is interested. These specifications are referred to as replication subscriptions. In addition the replication database and server must be configured to support the potentially large volumes of data that will be transferred between the source and recipient databases. Each of these important parameters is outlined in detail below.

## **7.2 Replication Definitions**

Replication definitions that have been defined against DM tables and columns are detailed herein.

TBS

## **7.3 Replication Subscriptions**

Replication subscriptions that have been defined against DM tables and columns are detailed herein.

TBS

## **7.4 Replication Database Configuration**

Replication Database Configuration specifications applicable to DM replication are contained herein.

TBS

## **7.5 Replication Server Configuration**

Replication Server Configuration specifications applicable to DM replication are contained herein.

TBS

This page intentionally left blank.

## **8. Scripts**

---

### **8.1 Installation Scripts**

Any scripts used to support installation of the DM database are described herein. These files may be found in the directory /ecs/formal/DM/DDICT/src/database.

Script File	Description
DmDdCreateAllRun.csh	Installs/populates Data Management database

### **8.2 De-Installation Scripts**

Any scripts used to support de-installation of the DM database are described herein.

TBS

### **8.3 Backup/Recovery Scripts**

Any scripts used to facilitate backup or recovery of the DM database are described herein.

TBS

### **8.4 Miscellaneous Scripts**

Miscellaneous scripts applicable to the DM database are described herein.

TBS

This page intentionally left blank.

## Abbreviations and Acronyms

---

ACL	Access Control List
ACMHW	Access and Control Management HWCI
ADC	affiliated data center
ADSHW	Advertising Server HWCI
ADSRV	Advertising Service CSCI
AI&T	algorithm integration and test
AITHW	Algorithm Integration and Test HWCI
AITTL	Algorithm Integration and Test CSCI
AM-1	EOS AM Project spacecraft 1, morning spacecraft series -- ASTER, CERES, MISR, MODIS and MOPITT instruments
ANSI	American National Standards Institute
API	application program (or programming) interface
APID	application's process ID
AQAHW	Algorithm QA HWCI
ASCII	American Standard Code for Information Exchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer (formerly ITIR)
AVHRR	Advanced Very High-Resolution Radiometer
BER	bit error rate
BUFR	binary universal format for representation of data
CASE	Computer Aided Software Engineering
CCSDS	Consultative Committee for Space Data Systems
CD	contractual delivery 214-001
CD-ROM	compact disk -- read only memory
CDR	Critical Design Review
CDRL	contract data requirements list
CERES	Clouds and Earth's Radiant Energy System

CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CPU	central processing unit
CSCI	computer software configuration item
CSDT	Computer Science Data Type
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem
DAAC	Distributed Active Archive Center
DAN	data availability notice
DAO	Data Assimilation Office
DAR	data acquisition request
DAS	data availability schedule
DBMS	Database Management System
DDICT	Data Dictionary CSCI
DDIST	Data Distribution Services CSCI
DDSRV	Document Data Server CSCI
DESKT	Desktop CSCI
DID	data item description
DIM	distributed information manager (SDPS)
DIMGR	Distributed Information Manager CSCI
DIPHW	Distribution and Ingest Peripheral Management HWCI
DMGHW	Data Management HWCI
DMS	Data Management Subsystem
DMWG	Data Management Working Group
DP	Data Provider
DPR	data processing request
DPREP	Science Data Preprocessing CSCI
DPS	Data Processing Subsystem
DRPHW	Data Repository HWCI

DSS	Data Server Subsystem
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EDOS	EOS Data and Operations System
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing (Descending) Mission -- see AM-1
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FDDI	fiber distributed data interface
FDF	flight dynamics facility
FDFEPHEM	FDF-generated definitive orbit data
FGDC	Federal Geographic Data Committee
FK	Foreign Key
FOO	Flight of Opportunity
FOS	Flight Operations Segment (ECS)
GB	gigabyte ( $10^9$ )
GNU	(recursive acronym: “GNU’s Not Unix”); a project supported by the Free Software Foundation dedicated to the delivery of free software
GPCP	Global Precipitation Climatology Project
GPCP	Global Precipitation Climatology Project
GPI	GOES Precipitation Index
GRIB	GRid In Binary
GSFC	Goddard Space Flight Center
GTWAY	Version 0 Interoperability Gateway CSCI
GUI	graphic user interface

GV	ground validation
HDF	hierarchical data format
HDF-EOS	an EOS proposed standard for a specialized HDF data format
HIPPI	high performance parallel interface
HMI	human machine interface
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
HWCI	hardware configuration item
I&T	integration and test
I/F	interface
I/O	input/output
ICD	interface control document
ICLHW	Ingest Client HWCI
ID	identification
IDE	Interactive Development Environments
IDG	Infrastructure Development Group
IDR	Incremental Design Review
IERS	International Earth Rotation Service
IMS	Information Management System (obsolete ECS element name)
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
IP	international partners
IR-1	Interim Release 1
IRD	interface requirements document
ISO	International Standards Organization
ISS	Internetworking Subsystem
IV&V	independent verification and validation
JPL	Jet Propulsion Laboratory
L0-L4	Level 0 (zero) through Level 4

LaRC	Langley Research Center (DAAC)
LIM	local information manager (SDPS)
LIMGR	Local Information Manager CSCI
LIS	Lightning Imaging Sensor
LSM	local system management (ECS)
MB	megabyte ( $10^6$ )
MDT	mean downtime
MDT	mean downtime
MFLOPS	mega (millions of) floating-point operations ( $10^6$ ) per second
MISR	Multi-Angle Imaging SpectroRadiometer
MODIS	Moderate-Resolution Imaging Spectrometer
MOPITT	Measurements of Pollution in the Troposphere
MSFC	Marshall Space Flight Center
MSS	Management Support Subsystem
MTBF	mean time between failure
MTPE	Mission to Planet Earth
MTTR	mean time to restore
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NESDIS	National Environmental Satellite Data and Information Service
NMC	National Meteorological Center (NOAA)
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
O/A	orbit/altitude
ODC	other data center
OSI	Open System Interconnect
PDPS	Planning and Data Processing Subsystem
PDR	Preliminary Design Review

PDS	production data set
PGE	Product Generation Executive
PGS	Product Generation System (obsolete ECS element name) (ASTER)
PK	Primary Key
PLANG	Production Planning CSCI
PLNHW	Planning HWCI
PLS	Planning Subsystem
POSIX	Portable Operating System Interface for Computer Environments
PR	Precipitation Radar (TRMM)
PRONG	Processing CSCI
QA	quality assurance
RMA	reliability, maintainability, availability
RTF	rich text format
SAA	satellite active archive
SAGE	Stratospheric Aerosol and Gas Experiment
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility (GSFC)
SDPS	Science Data Processing Segment (ECS)
SDPTK	SDP Toolkit CSCI
SDSRV	Science Data Server CSCI
SeaWIFS II	Sea-Viewing Wide Field-of-View Sensor II
SFDU	Standard Format Data Unit
SMC	System Management Center (ECS)
SPRHW	Science Processing HWCI
SRS	software requirements specification
SSM/I	Special Sensor for Microwave/Imaging (DMSP)
SST	sea surface temperature
STMGMT	Storage Management

STMGT	Storage Management Software CSCI
SUBSRV	Subscription Server
TMI	TRMM Microwave Image
TOMS	Total Ozone Mapping Spectrometer
TONS	TDRS On-board Navigational System
TRMM	Tropical Rainfall Measuring Mission (joint US-Japan)
TSDIS	TRMM Science Data and Information System
USNO	US Naval Observatory
UT	universal time
UTC	universal time code
V0	Version 0
VIRS	Visible Infrared Scanner (TRMM)
WAIS	Wide Area Information Server
WKBCH	Workbench CSCI
WKSHW	Working Storage HWCI
WWW	World-Wide Web

This page intentionally left blank.